US 20250258655A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0258655 A1**

**Hagmajer** (43) **Pub. Date: Aug. 14, 2025**

(54) **METHODS AND SYSTEMS FOR SOFTWARE CODE GENERATION**

(71) Applicant: **Box, Inc.**, Redwood City, CA (US)

(72) Inventor: **Marcin Marek Hagmajer**, Warszawa (PL)

(21) Appl. No.: **18/439,158**

(22) Filed: **Feb. 12, 2024**

**Publication Classification**

(51) **Int. Cl.**

***G06F 8/30*** (2018.01)

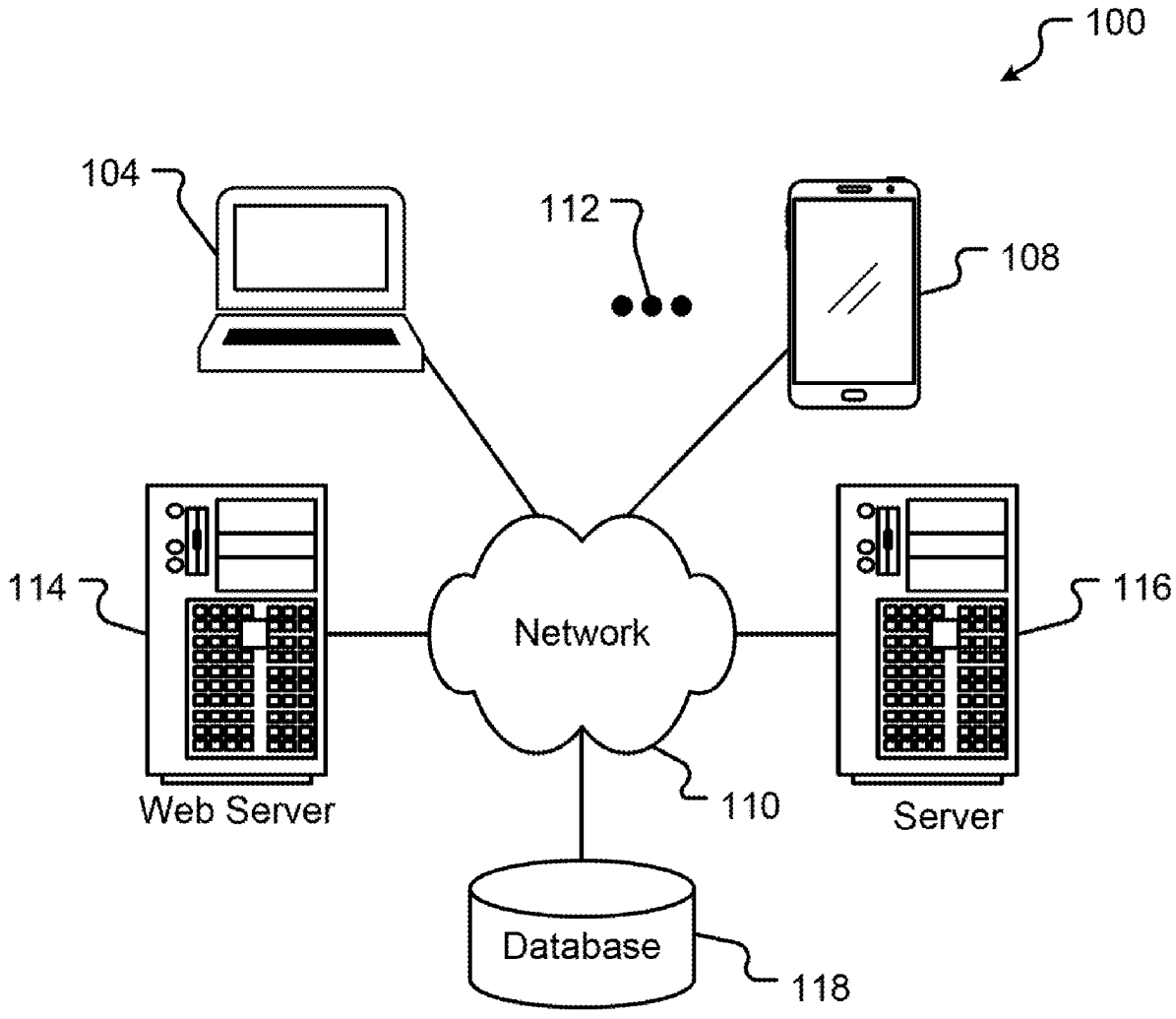
***G06F 8/41*** (2018.01)

(52) **U.S. Cl.**

CPC . ***G06F 8/30*** (2013.01); ***G06F 8/41*** (2013.01)

(57) **ABSTRACT**

Embodiments of the disclosure provide systems and methods for automated generation of computer source code across multiple programming languages. According to one embodiment, a method for automated generation of computer source code can comprise receiving information defining a coding node. The coding node can comprise a definition of an object within an object tree for the computer source code. The coding node can be created based on the received information defining the coding node and the coding node can be transformed into an intermediate, transformed coding node. The intermediate, transformed coding node can then be transpiled into software source code in a target programming language of a plurality of programming languages.



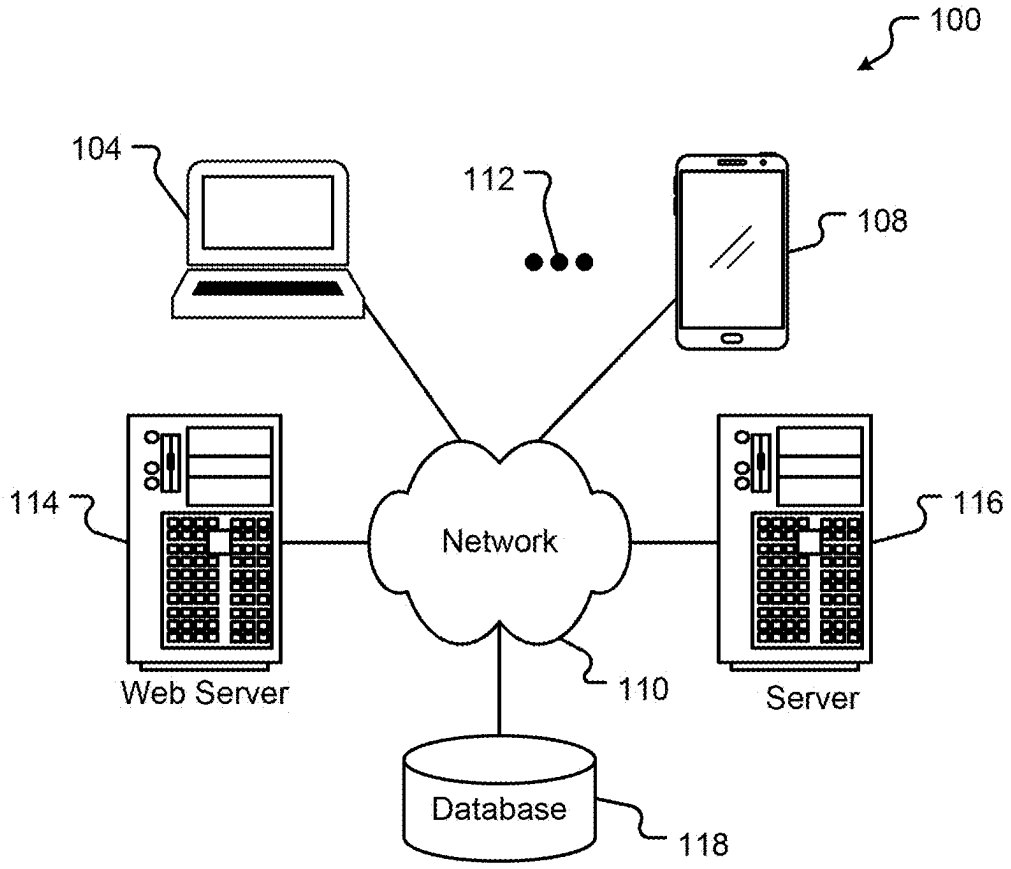

100
104
112
108
114
Network
116
Web Server
110
Server
Database
118

*FIG. 1*

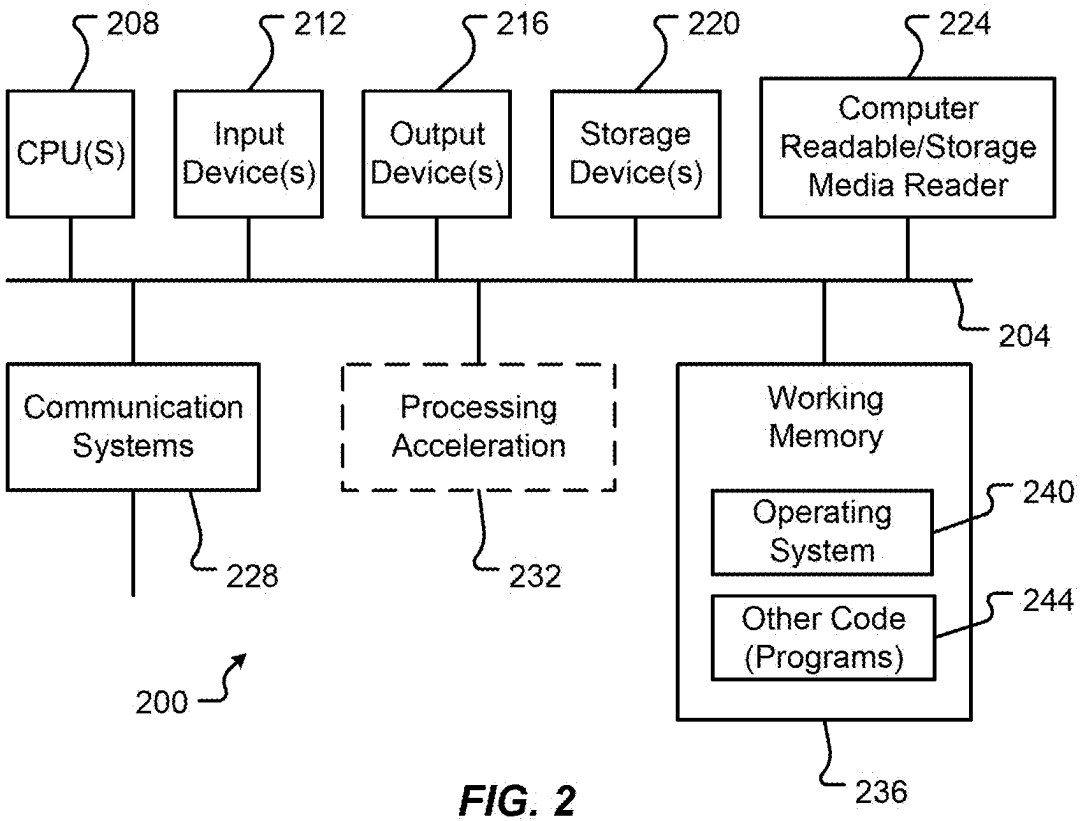

208
212
216
220
224
CPU(S)
Input Device(s)
Output Device(s)
Storage Device(s)
Computer Readable/Storage Media Reader
204
Communication Systems
Processing Acceleration
Working Memory
Operating System
Other Code (Programs)
240
244
228
232
200
236

*FIG. 2*

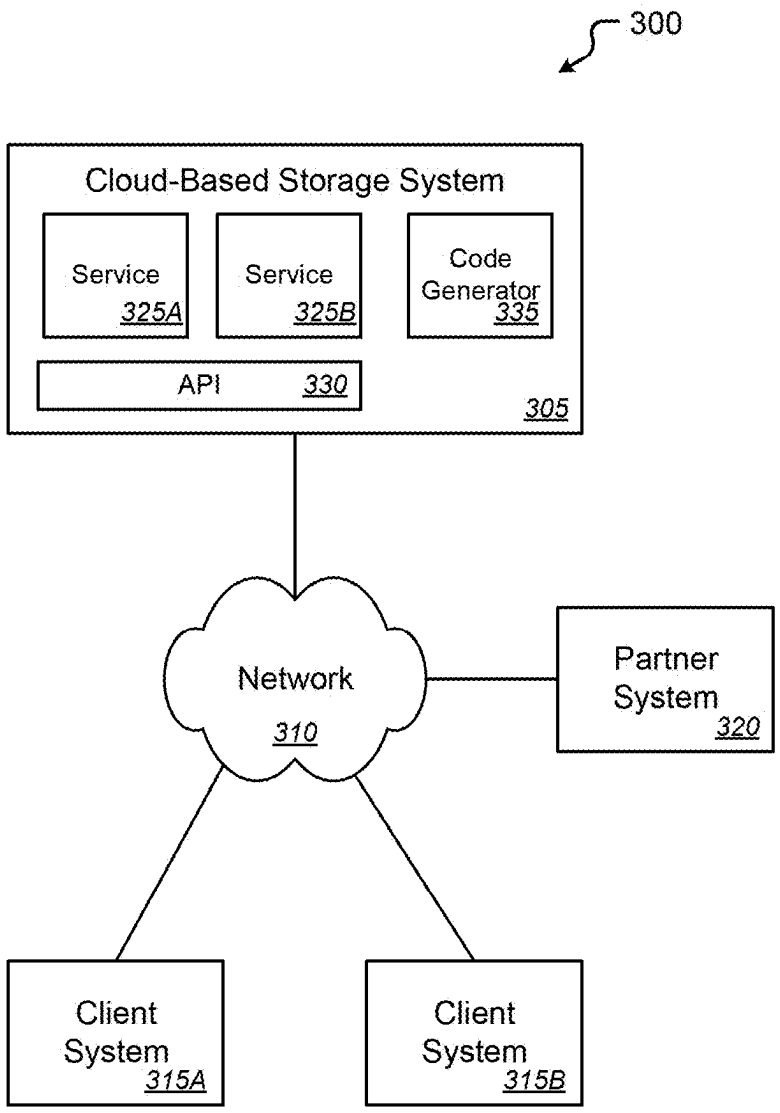

300
Cloud-Based Storage System
Service
325A
Service
325B
Code
Generator
335
API
330
305
Network
310
Partner
System
320
Client
System
315A
Client
System
315B

*FIG. 3*

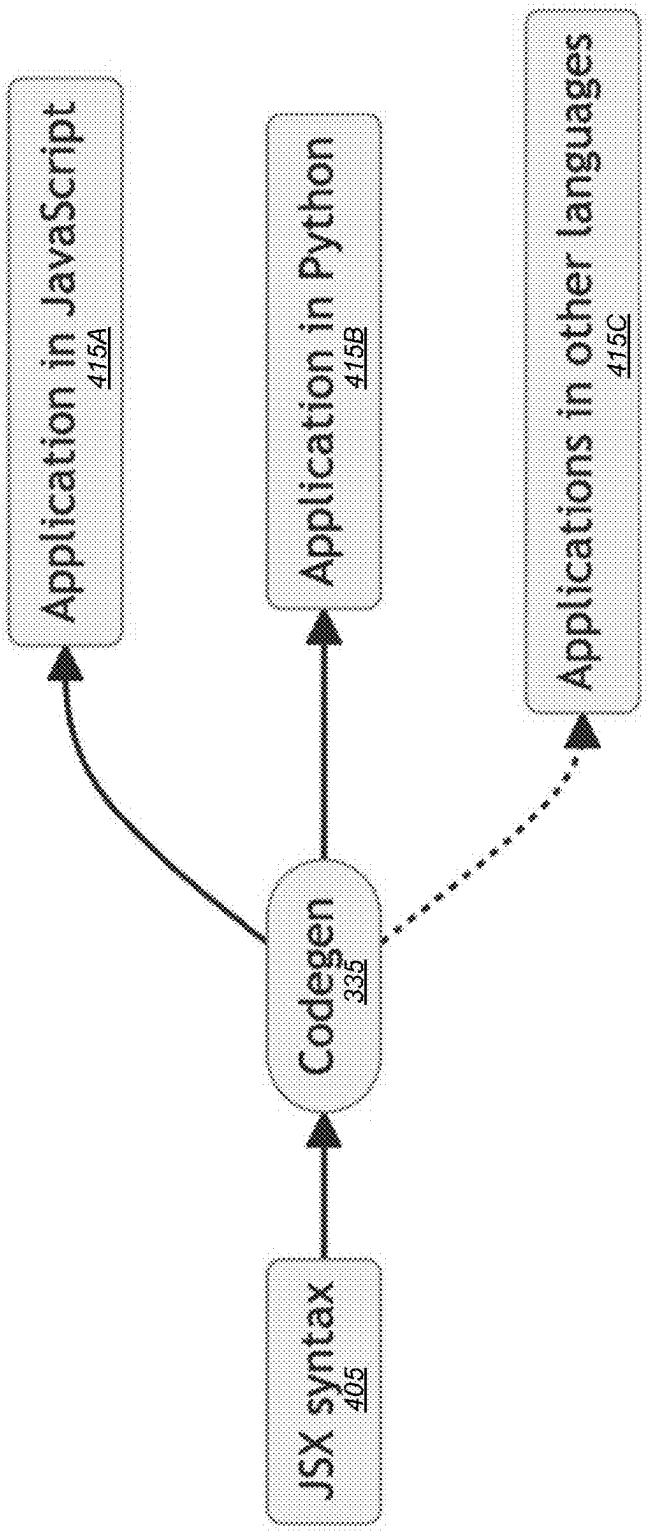

JSX syntax
405
Codegen
335
Application in JavaScript
415A
Application in Python
415B
Applications in other languages
415C

FIG. 4

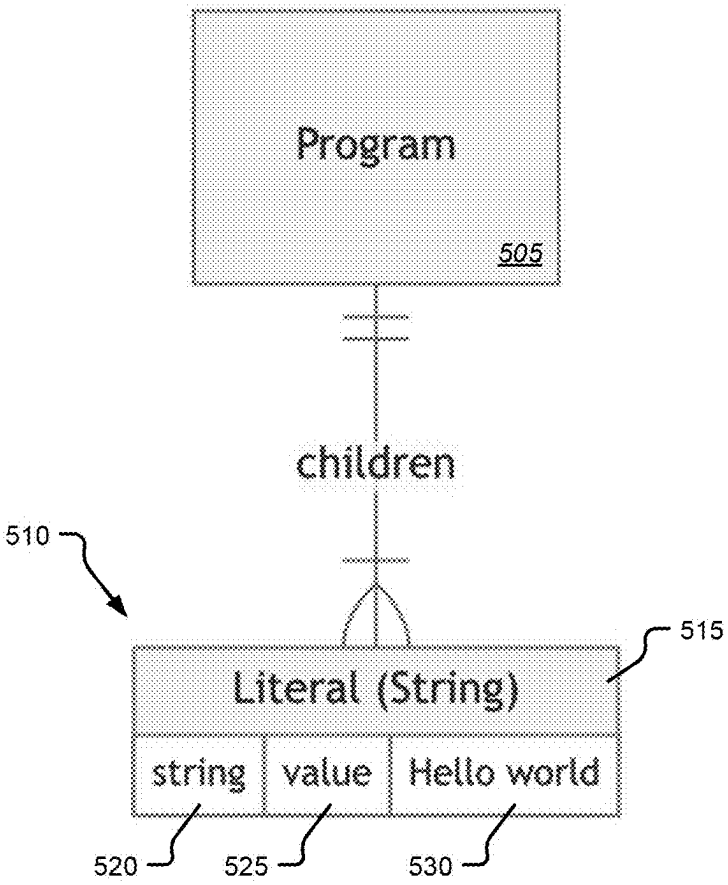

Program
505
children
510
515
Literal (String)
string
value
Hello world
520
525
530

*FIG. 5*

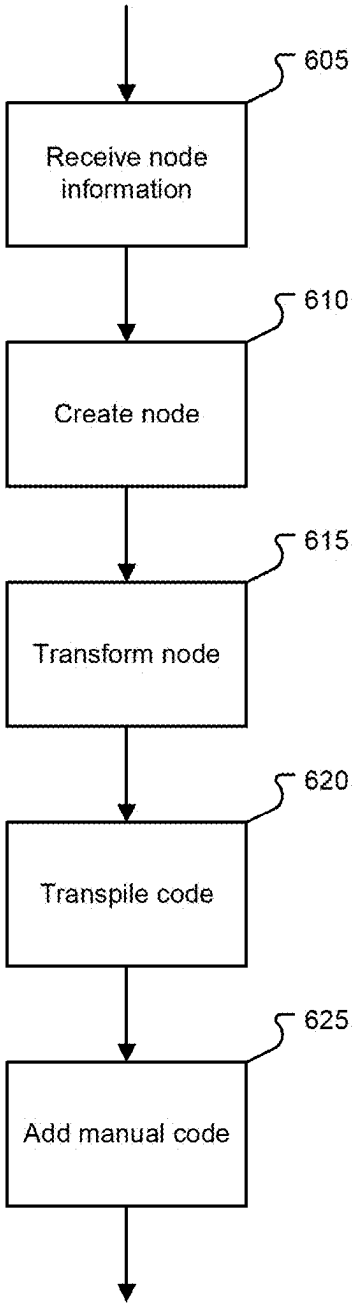

605
Receive node information
610
Create node
615
Transform node
620
Transpile code
625
Add manual code

*Fig. 6*

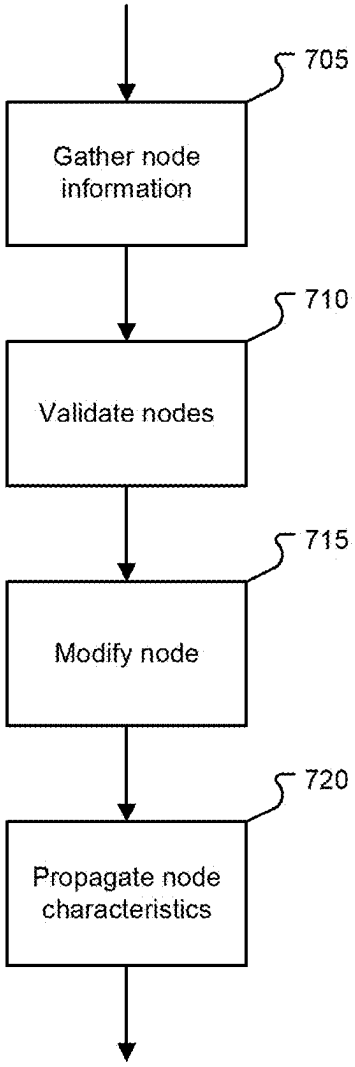

705
Gather node information
710
Validate nodes
715
Modify node
720
Propagate node characteristics

*Fig. 7*

# METHODS AND SYSTEMS FOR SOFTWARE CODE GENERATION

## FIELD OF THE DISCLOSURE

**[0001]** Embodiments of the present disclosure relate generally to methods and systems for computer programming and more particularly to automated generation of computer source code across multiple programming languages.

## BACKGROUND

**[0002]** The tech industry has long grappled with the inefficiencies associated with developing software for multiple platforms. By necessitating manual reimplementation across diverse programming languages, companies face inflated costs and risks of inconsistency. Hence, there is a need for improved methods and systems for automated and uniform code generation, reducing the maintenance burden.

## BRIEF SUMMARY

**[0003]** Embodiments of the disclosure provide systems and methods for automated generation of computer source code across multiple programming languages. According to one embodiment, a method for automated generation of computer source code can comprise receiving information defining a coding node. The coding node can comprise a definition of an object within an object tree for the computer source code. According to one embodiment, the information defining the coding node can comprise Javascript Syntax extension (JSX) code. Additionally, or alternatively, the information defining the coding node can comprise user defined information. The languages composed with the coding node types is Turing-complete. Hence, it can be used to expressed any algorithm.

**[0004]** The coding node can be created based on the received information defining the coding node and the coding node can be transformed into an intermediate, transformed coding node. Creating the coding node can comprise generating one or more of: a node type; one or more node attributes; or one or more child nodes for the coding node based on the received information defining the coding node. In some cases, creating the coding node can further comprise performing a preliminary validation on the coding node. Transforming the coding node into the intermediate, transformed coding node can comprise gathering additional information defining a context for the coding node, validating the coding node for unresolved references and anomalies, modifying the coding node based on the information defining the coding node and the object tree, and propagating one or more code characteristics from the modified coding node to the intermediate, transformed coding node.

**[0005]** The intermediate, transformed coding node can then be transpiled into software source code in a target programming language of a plurality of programming languages. Transpiling the intermediate, transformed coding node into the source code in the target programming language can comprise selecting a transpiler from a plurality of available transpilers based on the target programming language.

**[0006]** According to another embodiment, a system can comprise a processor and a memory coupled with and readable by the processor. The memory can store therein a set of instructions which, when executed by the processor, causes the processor to receive information defining a coding node. The coding node can comprise a definition of an object within an object tree for the computer source code. According to one embodiment, the information defining the coding node can comprise Javascript Syntax extension (JSX) code. Additionally, or alternatively, the information defining the coding node can comprise user defined information.

**[0007]** The instructions can further cause the processor to create the coding node based on the received information defining the coding node and transform the coding node into an intermediate, transformed coding node. Creating the coding node can comprise generating one or more of: a node type; one or more node attributes; or one or more child nodes for the coding node based on the received information defining the coding node. In some cases, creating the coding node can further comprise performing a preliminary validation on the coding node. Transforming the coding node into the intermediate, transformed coding node can comprise gathering additional information defining a context for the coding node, validating the coding node for unresolved references and anomalies, modifying the coding node based on the information defining the coding node and the object tree, and propagating one or more code characteristics from the modified coding node to the intermediate, transformed coding node.

**[0008]** The instructions can further cause the processor to transpile the intermediate, transformed coding node into software source code in a target programming language of a plurality of programming languages. Transpiling the intermediate, transformed coding node into the source code in the target programming language can comprise selecting a transpiler from a plurality of available transpilers based on the target programming language.

**[0009]** According to yet another embodiment, a non-transitory, computer-readable medium can comprise a set of instructions stored therein which, when executed by a processor, causes the processor to receive information defining a coding node. The coding node can comprise a definition of an object within an object tree for the computer source code. According to one embodiment, the information defining the coding node can comprise Javascript Syntax extension (JSX) code. Additionally, or alternatively, the information defining the coding node can comprise user defined information.

**[0010]** The instructions can further cause the processor to create the coding node based on the received information defining the coding node and transform the coding node into an intermediate, transformed coding node. Creating the coding node can comprise generating one or more of: a node type; one or more node attributes; or one or more child nodes for the coding node based on the received information defining the coding node. In some cases, creating the coding node can further comprise performing a preliminary validation on the coding node. Transforming the coding node into the intermediate, transformed coding node can comprise gathering additional information defining a context for the coding node, validating the coding node for unresolved references and anomalies, modifying the coding node based on the information defining the coding node and the object tree, and propagating one or more code characteristics from the modified coding node to the intermediate, transformed coding node.

**[0011]** The instructions can further cause the processor to transpile the intermediate, transformed coding node into

software source code in a target programming language of a plurality of programming languages.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0012]** FIG. **1** is a block diagram illustrating elements of an exemplary computing environment in which embodiments of the present disclosure may be implemented.

**[0013]** FIG. **2** is a block diagram illustrating elements of an exemplary computing device in which embodiments of the present disclosure may be implemented.

**[0014]** FIG. **3** is a block diagram illustrating an exemplary environment in which automated code generation can be performed according to one embodiment of the present disclosure.

**[0015]** FIG. **4** is a block diagram conceptually illustrating automated code generation according to one embodiment of the present disclosure.

**[0016]** FIG. **5** is a block diagram illustrating content of a coding node according to one embodiment of the present disclosure.

**[0017]** FIG. **6** is a flowchart illustrating an exemplary process for code generation according to one embodiment of the present disclosure.

**[0018]** FIG. **7** is a flowchart illustrating an exemplary process for coding node transformation according to one embodiment of the present disclosure.

**[0019]** In the appended figures, similar components and/or features may have the same reference label. Further, various components of the same type may be distinguished by following the reference label by a letter that distinguishes among the similar components. If only the first reference label is used in the specification, the description is applicable to any one of the similar components having the same first reference label irrespective of the second reference label.

## DETAILED DESCRIPTION

**[0020]** In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of various embodiments disclosed herein. It will be apparent, however, to one skilled in the art that various embodiments of the present disclosure may be practiced without some of these specific details. The ensuing description provides exemplary embodiments only and is not intended to limit the scope or applicability of the disclosure. Furthermore, to avoid unnecessarily obscuring the present disclosure, the preceding description omits a number of known structures and devices. This omission is not to be construed as a limitation of the scopes of the claims. Rather, the ensuing description of the exemplary embodiments will provide those skilled in the art with an enabling description for implementing an exemplary embodiment. It should however be appreciated that the present disclosure may be practiced in a variety of ways beyond the specific detail set forth herein.

**[0021]** While the exemplary aspects, embodiments, and/or configurations illustrated herein show the various components of the system collocated, certain components of the system can be located remotely, at distant portions of a distributed network, such as a Local-Area Network (LAN) and/or Wide-Area Network (WAN) such as the Internet, or within a dedicated system. Thus, it should be appreciated, that the components of the system can be combined in to one or more devices or collocated on a particular node of a distributed network, such as an analog and/or digital telecommunications network, a packet-switch network, or a circuit-switched network. It will be appreciated from the following description, and for reasons of computational efficiency, that the components of the system can be arranged at any location within a distributed network of components without affecting the operation of the system.

**[0022]** Furthermore, it should be appreciated that the various links connecting the elements can be wired or wireless links, or any combination thereof, or any other known or later developed element(s) that is capable of supplying and/or communicating data to and from the connected elements. These wired or wireless links can also be secure links and may be capable of communicating encrypted information. Transmission media used as links, for example, can be any suitable carrier for electrical signals, including coaxial cables, copper wire and fiber optics, and may take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

**[0023]** As used herein, the phrases "at least one," "one or more," "or," and "and/or" are open-ended expressions that are both conjunctive and disjunctive in operation. For example, each of the expressions "at least one of A, B and C," "at least one of A, B, or C," "one or more of A, B, and C," "one or more of A, B, or C," "A, B, and/or C," and "A, B, or C" means A alone, B alone, C alone, A and B together, A and C together, B and C together, or A, B and C together.

**[0024]** The term "a" or "an" entity refers to one or more of that entity. As such, the terms "a" (or "an"), "one or more" and "at least one" can be used interchangeably herein. It is also to be noted that the terms "comprising," "including," and "having" can be used interchangeably.

**[0025]** The term "automatic" and variations thereof, as used herein, refers to any process or operation done without material human input when the process or operation is performed. However, a process or operation can be automatic, even though performance of the process or operation uses material or immaterial human input, if the input is received before performance of the process or operation. Human input is deemed to be material if such input influences how the process or operation will be performed. Human input that consents to the performance of the process or operation is not deemed to be "material."

**[0026]** The term "computer-readable medium" as used herein refers to any tangible storage and/or transmission medium that participate in providing instructions to a processor for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, Non-Volatile Random-Access Memory (NVRAM), or magnetic or optical disks. Volatile media includes dynamic memory, such as main memory. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, magneto-optical medium, a Compact Disk Read-Only Memory (CD-ROM), any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a Random-Access Memory (RAM), a Programmable Read-Only Memory (PROM), and Erasable Programable Read-Only Memory (EPROM), a Flash-EPROM, a solid state medium like a memory card, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read. A digital file attachment to

e-mail or other self-contained information archive or set of archives is considered a distribution medium equivalent to a tangible storage medium. When the computer-readable media is configured as a database, it is to be understood that the database may be any type of database, such as relational, hierarchical, object-oriented, and/or the like. Accordingly, the disclosure is considered to include a tangible storage medium or distribution medium and prior art-recognized equivalents and successor media, in which the software implementations of the present disclosure are stored.

**[0027]** A "computer readable signal" medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electromagnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device. Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, Radio Frequency (RF), etc., or any suitable combination of the foregoing.

**[0028]** The terms "determine," "calculate," and "compute," and variations thereof, as used herein, are used interchangeably and include any type of methodology, process, mathematical operation or technique.

**[0029]** It shall be understood that the term "means" as used herein shall be given its broadest possible interpretation in accordance with 35 U.S.C., Section 112, Paragraph 6. Accordingly, a claim incorporating the term "means" shall cover all structures, materials, or acts set forth herein, and all of the equivalents thereof. Further, the structures, materials or acts and the equivalents thereof shall include all those described in the summary of the disclosure, brief description of the drawings, detailed description, abstract, and claims themselves.

**[0030]** Aspects of the present disclosure may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, microcode, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium.

**[0031]** In yet another embodiment, the systems and methods of this disclosure can be implemented in conjunction with a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit element(s), an ASIC or other integrated circuit, a digital signal processor, a hard-wired electronic or logic circuit such as discrete element circuit, a programmable logic device or gate array such as Programmable Logic Device (PLD), Programmable Logic Array (PLA), Field Programmable Gate Array (FPGA), Programmable Array Logic (PAL), special purpose computer, any comparable means, or the like. In general, any device(s) or means capable of implementing the methodology illustrated herein can be used to implement the various aspects of this disclosure. Exemplary hardware that can be used for the disclosed embodiments, configurations, and aspects includes computers, handheld devices, telephones (e.g., cellular, Internet enabled, digital, analog, hybrids, and others), and other hardware known in the art. Some of these devices include processors (e.g., a single or multiple microprocessors), memory, nonvolatile storage, input devices, and output devices. Furthermore, alternative software implementations including, but not limited to, distributed processing or component/object distributed processing, parallel processing, or virtual machine processing can also be constructed to implement the methods described herein.

**[0032]** Examples of the processors as described herein may include, but are not limited to, at least one of Qualcomm® Snapdragon® 800 and 801, Qualcomm® Snapdragon® 610 and 615 with 4G LTE Integration and 64-bit computing, Apple® A7 processor with 64-bit architecture, Apple® M7 motion coprocessors, Samsung® Exynos® series, the Intel® Core™ family of processors, the Intel® Xeon® family of processors, the Intel® Atom™ family of processors, the Intel Itanium® family of processors, Intel® Core® i5-4670K and i7-4770K 22nm Haswell, Intel® Core® i5-3570K 22nm Ivy Bridge, the AMD® FX™ family of processors, AMD® FX-4300, FX-6300, and FX-8350 32nm Vishera, AMD® Kaveri processors, Texas Instruments® Jacinto C6000™ automotive infotainment processors, Texas Instruments® OMAP™ automotive-grade mobile processors, ARM® Cortex™-M processors, ARM® Cortex-A and ARM926EJ-S™ processors, other industry-equivalent processors, and may perform computational functions using any known or future-developed standard, instruction set, libraries, and/or architecture.

**[0033]** In yet another embodiment, the disclosed methods may be readily implemented in conjunction with software using object or object-oriented software development environments that provide portable source code that can be used on a variety of computer or workstation platforms. Alternatively, the disclosed system may be implemented partially or fully in hardware using standard logic circuits or Very Large-Scale Integration (VLSI) design. Whether software or hardware is used to implement the systems in accordance with this disclosure is dependent on the speed and/or efficiency requirements of the system, the particular function, and the particular software or hardware systems or microprocessor or microcomputer systems being utilized.

**[0034]** In yet another embodiment, the disclosed methods may be partially implemented in software that can be stored on a storage medium, executed on programmed general-purpose computer with the cooperation of a controller and memory, a special purpose computer, a microprocessor, or the like. In these instances, the systems and methods of this disclosure can be implemented as program embedded on personal computer such as an applet, JAVA® or Common Gateway Interface (CGI) script, as a resource residing on a server or computer workstation, as a routine embedded in a dedicated measurement system, system component, or the like. The system can also be implemented by physically incorporating the system and/or method into a software and/or hardware system.

**[0035]** Although the present disclosure describes components and functions implemented in the aspects, embodiments, and/or configurations with reference to particular standards and protocols, the aspects, embodiments, and/or configurations are not limited to such standards and protocols. Other similar standards and protocols not mentioned

herein are in existence and are considered to be included in the present disclosure. Moreover, the standards and protocols mentioned herein and other similar standards and protocols not mentioned herein are periodically superseded by faster or more effective equivalents having essentially the same functions. Such replacement standards and protocols having the same functions are considered equivalents included in the present disclosure.

**[0036]** Various additional details of embodiments of the present disclosure will be described below with reference to the figures. While the flowcharts will be discussed and illustrated in relation to a particular sequence of events, it should be appreciated that changes, additions, and omissions to this sequence can occur without materially affecting the operation of the disclosed embodiments, configuration, and aspects.

**[0037]** FIG. **1** is a block diagram illustrating elements of an exemplary computing environment in which embodiments of the present disclosure may be implemented. More specifically, this example illustrates a computing environment **100** that may function as the servers, user computers, or other systems provided and described herein. The environment **100** includes one or more user computers, or computing devices, such as a computing device **104**, a communication device **108**, and/or more **112**. The computing devices **104**, **108**, **112** may include general purpose personal computers (including, merely by way of example, personal computers, and/or laptop computers running various versions of Microsoft Corp.'s Windows® and/or Apple Corp.'s Macintosh® operating systems) and/or workstation computers running any of a variety of commercially-available UNIX® or UNIX-like operating systems. These computing devices **104**, **108**, **112** may also have any of a variety of applications, including for example, database client and/or server applications, and web browser applications. Alternatively, the computing devices **104**, **108**, **112** may be any other electronic device, such as a thin-client computer, Internet-enabled mobile telephone, and/or personal digital assistant, capable of communicating via a network **110** and/or displaying and navigating web pages or other types of electronic documents. Although the exemplary computer environment **100** is shown with two computing devices, any number of user computers or computing devices may be supported.

**[0038]** Environment **100** further includes a network **110**. The network **110** may can be any type of network familiar to those skilled in the art that can support data communications using any of a variety of commercially-available protocols, including without limitation Session Initiation Protocol (SIP), Transmission Control Protocol/Internet Protocol (TCP/IP), Systems Network Architecture (SNA), Internetwork Packet Exchange (IPX), AppleTalk, and the like. Merely by way of example, the network **110** maybe a Local Area Network (LAN), such as an Ethernet network, a Token-Ring network and/or the like; a wide-area network; a virtual network, including without limitation a Virtual Private Network (VPN); the Internet; an intranet; an extranet; a Public Switched Telephone Network (PSTN); an infra-red network; a wireless network (e.g., a network operating under any of the IEEE 802.9 suite of protocols, the Bluetooth® protocol known in the art, and/or any other wireless protocol); and/or any combination of these and/or other networks.

**[0039]** The system may also include one or more servers **114**, **116**. In this example, server **114** is shown as a web server and server **116** is shown as an application server. The web server **114**, which may be used to process requests for web pages or other electronic documents from computing devices **104**, **108**, **112**. The web server **114** can be running an operating system including any of those discussed above, as well as any commercially-available server operating systems. The web server **114** can also run a variety of server applications, including SIP servers, HyperText Transfer Protocol (secure) (HTTP(s)) servers, FTP servers, CGI servers, database servers, Java servers, and the like. In some instances, the web server **114** may publish operations available operations as one or more web services.

**[0040]** The environment **100** may also include one or more file and or/application servers **116**, which can, in addition to an operating system, include one or more applications accessible by a client running on one or more of the computing devices **104**, **108**, **112**. The server(s) **116** and/or **114** may be one or more general purpose computers capable of executing programs or scripts in response to the computing devices **104**, **108**, **112**. As one example, the server **116**, **114** may execute one or more web applications. The web application may be implemented as one or more scripts or programs written in any programming language, such as Java™ , C, C#®, or C++, and/or any scripting language, such as Perl, Python, or Tool Command Language (TCL), as well as combinations of any programming/scripting languages. The application server(s) **116** may also include database servers, including without limitation those commercially available from Oracle®, Microsoft®, Sybase®, IBM® and the like, which can process requests from database clients running on a computing device **104**, **108**, **112**.

**[0041]** The web pages created by the server **114** and/or **116** may be forwarded to a computing device **104**, **108**, **112** via a web (file) server **114**, **116**. Similarly, the web server **114** may be able to receive web page requests, web services invocations, and/or input data from a computing device **104**, **108**, **112** (e.g., a user computer, etc.) and can forward the web page requests and/or input data to the web (application) server **116**. In further embodiments, the server **116** may function as a file server. Although for ease of description, FIG. **1** illustrates a separate web server **114** and file/application server **116**, those skilled in the art will recognize that the functions described with respect to servers **114**, **116** may be performed by a single server and/or a plurality of specialized servers, depending on implementation-specific needs and parameters. The computer systems **104**, **108**, **112**, web (file) server **114** and/or web (application) server **116** may function as the system, devices, or components described herein.

**[0042]** The environment **100** may also include a database **118**. The database **118** may reside in a variety of locations. By way of example, database **118** may reside on a storage medium local to (and/or resident in) one or more of the computers **104**, **108**, **112**, **114**, **116**. Alternatively, it may be remote from any or all of the computers **104**, **108**, **112**, **114**, **116**, and in communication (e.g., via the network **110**) with one or more of these. The database **118** may reside in a Storage-Area Network (SAN) familiar to those skilled in the art. Similarly, any necessary files for performing the functions attributed to the computers **104**, **108**, **112**, **114**, **116** may be stored locally on the respective computer and/or remotely, as appropriate. The database **118** may be a relational database, such as Oracle 20i®, that is adapted to store,

update, and retrieve data in response to Structured Query Language (SQL) formatted commands.

**[0043]** FIG. **2** is a block diagram illustrating elements of an exemplary computing device in which embodiments of the present disclosure may be implemented. More specifically, this example illustrates one embodiment of a computer system **200** upon which the servers, user computers, computing devices, or other systems or components described above may be deployed or executed. The computer system **200** is shown comprising hardware elements that may be electrically coupled via a bus **204**. The hardware elements may include one or more Central Processing Units (CPUs) **208**; one or more input devices **212** (e.g., a mouse, a keyboard, etc.); and one or more output devices **216** (e.g., a display device, a printer, etc.). The computer system **200** may also include one or more storage devices **220**. By way of example, storage device(s) **220** may be disk drives, optical storage devices, solid-state storage devices such as a Random-Access Memory (RAM) and/or a Read-Only Memory (ROM), which can be programmable, flash-updateable and/or the like.

**[0044]** The computer system **200** may additionally include a computer-readable storage media reader **224**; a communications system **228** (e.g., a modem, a network card (wireless or wired), an infra-red communication device, etc.); and working memory **236**, which may include RAM and ROM devices as described above. The computer system **200** may also include a processing acceleration unit **232**, which can include a Digital Signal Processor (DSP), a special-purpose processor, and/or the like.

**[0045]** The computer-readable storage media reader **224** can further be connected to a computer-readable storage medium, together (and, optionally, in combination with storage device(s) **220**) comprehensively representing remote, local, fixed, and/or removable storage devices plus storage media for temporarily and/or more permanently containing computer-readable information. The communications system **228** may permit data to be exchanged with a network and/or any other computer described above with respect to the computer environments described herein. Moreover, as disclosed herein, the term "storage medium" may represent one or more devices for storing data, including ROM, RAM, magnetic RAM, core memory, magnetic disk storage mediums, optical storage mediums, flash memory devices and/or other machine-readable mediums for storing information.

**[0046]** The computer system **200** may also comprise software elements, shown as being currently located within a working memory **236**, including an operating system **240** and/or other code **244**. It should be appreciated that alternate embodiments of a computer system **200** may have numerous variations from that described above. For example, customized hardware might also be used and/or particular elements might be implemented in hardware, software (including portable software, such as applets), or both. Further, connection to other computing devices such as network input/output devices may be employed.

**[0047]** Examples of the processors 208 as described herein may include, but are not limited to, at least one of Qualcomm® Snapdragon® 800 and 801, Qualcomm® Snapdragon® 620 and 615 with 4G LTE Integration and 64-bit computing, Apple® A7 processor with 64-bit architecture, Apple® M7 motion coprocessors, Samsung® Exynos® series, the Intel® Core™ family of processors, the Intel® Xeon® family of processors, the Intel® Atom™ family of processors, the Intel Itanium® family of processors, Intel® Core® i5-4670K and i7-4770K 22nm Haswell, Intel® Core® i5-3570K 22nm Ivy Bridge, the AMD® FXT™ family of processors, AMD® FX-4300, FX-6300, and FX-8350 32nm Vishera, AMD® Kaveri processors, Texas Instruments® Jacinto C6000™ automotive infotainment processors, Texas Instruments® OMAP™ automotive-grade mobile processors, ARM® Cortex™-M processors, ARM® Cortex-A and ARM926EJ-S™ processors, other industry-equivalent processors, and may perform computational functions using any known or future-developed standard, instruction set, libraries, and/or architecture.

**[0048]** FIG. **3** is a block diagram illustrating an exemplary environment in which automated code generation can be performed according to one embodiment of the present disclosure. As illustrated in this example, the environment **300** can comprise a cloud-based storage system **305** coupled with a communications network **310**. The cloud-based storage system **305** can comprise any one or more servers and/or other computing devices as described above. Similarly, the communications network **310** can comprise any one or more wired and/or wireless, local-area and/or wide-area networks as described including, but not limited to, the Internet. Also coupled with the communications network **310** can be any one or more client devices **315**A and **315**B and/or one or more partner systems **320**. The one or more client devices **315**A and **315**B and/or one or more partner systems **320** can each comprise any one or more of the computing devices as described above.

**[0049]** Generally speaking, the cloud-based storage system **305** can provide any number of services **325**A and **325**B accessible by the one or more client devices **315**A and **315**B and/or one or more partner systems **320** via the communications network **310** and an Application Program Interface (API) **330**. Such service **325**A and **325**B can include, but are not limited to, services related to online storage and backup, an online collaboration environment, etc. In some cases, the one or more partner systems **320** can provide additional services to the client systems **315**A and **315**B based on and/or utilizing the services **325**A and **325**B provided by the cloud-based storage system.

**[0050]** According to one embodiment, the cloud-based storage system **305** can provide a code generator module **335**. As will be described below, the code generator module **335** can allow users of the partner systems **320** to easily and automatically generate code utilizing the one or more client devices **315**A and **315**B and/or one or more partner systems **320** regardless of the code languages utilized by the partner system **320**. For example, users of a partner system **320** can utilize the code generator module **335** to generate code to access and use the services **325**A and **325**B and the API **330** of the cloud-based storage system **305**. In other cases, the code generator module **335** can be used to generate code to accomplish any software purpose based on the input provided. For example, there can be a pricing computation algorithm of one partner company that takes a number of arguments and computes a relevant price. By representing the algorithm in JSX and the code generator module **335**, the partner company can assure that the algorithm works to consistently yield the same results on all target platforms.

**[0051]** FIG. **4** is a block diagram conceptually illustrating automated code generation according to one embodiment of the present disclosure. As illustrated in this example, auto-

mated generation of computer source code can comprise receiving or reading a JSX syntax code definition **405**. This code definition can define one or more coding nodes as will be described in further detail below with reference to FIG. **5**. Generally speaking, a code generator **335** can then transform and transpile this code definition **405** into any number, depending upon the number of programming languages supported, of computer application files **415**A-**415**C, e.g., source code and/or working code files in different programming languages. Additional details of generating computer source code as may be performed by the code generator **410** will be described in greater detail below with reference to FIG. **6**.

**[0052]** FIG. **5** is a block diagram illustrating content of a coding node according to one embodiment of the present disclosure. As illustrated in this example, a program **5050** can comprise one or more coding nodes **510**. The coding node, expressed in JSX syntax, for example, can comprise a type **515** for the node **510**. As illustrated in this example, the type **515** for the node **510** is "String" which is a subtype of "Literals" type. Other subtypes of Literals type in the reference implementation can be Boolean, Number, or Empty. The node **510** can further comprise a data type **520** for each of one or more attributes **525** and values **530** for the attributes **525**. Using the example illustrated here, the corresponding JSX syntax expression can be: <Program><String value="Hello world"/></Program>.

**[0053]** FIG. **6** is a flowchart illustrating an exemplary process for code generation according to one embodiment of the present disclosure. As illustrated in this example, automated generation of computer source code can comprise receiving **605** information defining a coding node. The coding node can comprise a definition of an object within an object tree for the computer source code. According to one embodiment, the information defining the coding node can comprise JSX code.

**[0054]** The coding node can be created **610** based on the received information defining the coding node. Creating **610** the coding node can comprise generating one or more of: a node type; one or more node attributes; or one or more child nodes for the coding node based on the received information defining the coding node. In some cases, creating **610** the coding node can further comprise performing a preliminary validation on the coding node.

**[0055]** The coding node can then be transformed **615** into an intermediate, transformed coding node. Generally speaking, intermediate, transformed coding nodes can act as a bridges between JSX constructs of the coding nodes and the final transpiled code. To do so, one or more transformers can deduct additional context, further validate, and adapt the coding nodes, with a focus on top-down, pre-order traversal. In some cases, a transformation accumulator can aid in complex transformations. Additional details of an exemplary transformation process will be described below with reference to FIG. **7**.

**[0056]** The intermediate, transformed coding node can then be transpiled **620** into software source code in a target programming language of a plurality of programming languages. Transpiling **620** the intermediate, transformed coding node into the source code in the target programming language can comprise selecting a transpiler from a plurality of available transpilers based on the target programming language.

**[0057]** In some cases, the information defining the coding node can further comprise manual code. Therefore, the processes can further comprise optionally adding **625** user defined, i.e., manually defined, information supplementing the JSX code. That is the code generator output is not limited to what can be expressed by the abstract JSX. It can also include platform specific code written in the target programming language to compose any software solution, i.e., partly generated, partly manual. In this way, users are not limited in any way. They can generate as much of the solution as practical and provide the rest as manual code together with the manual code definitions, e.g., for validation of coding nodes that depend on the manual code.

**[0058]** FIG. **7** is a flowchart illustrating an exemplary process for coding node transformation according to one embodiment of the present disclosure. As illustrated in this example, transforming the coding node into the intermediate, transformed coding node can comprise gathering **705** additional information defining a context for the coding node. The information provided here can come from the JSX syntax. It can include the type of the coding node, e.g., Program, If, Function, Literal, etc., its properties depending of the type of the node, e.g., name and arguments for a function, and children, i.e., the nested code nodes. Some properties can have default values assigned during creation in case they weren't included in the original JSX.

**[0059]** The coding node can then be validated **710** for unresolved references and anomalies. The validation of the node can depend on its type and can be performed to assure the node is used correctly. For example, a Program node cannot be used as a child of Function-it can only be used as a top-level node. Additional, or alternatively, validations can include checking if the values of properties are legal. For example, if use calls a function using Call node, a check can be made to determine if the number of provided arguments is correct.

**[0060]** The coding node can be modified **715** based on the information defining the coding node and the object tree. Based on the context of how a node is used and its contents, the transformation process may include modifying them. For example, if a Function accepts some complex data structure in its arguments, e.g., like a record of records, the transformer can extract it into its separate definitions so that the target programming language can define the structure first and then reference it. Some languages do not support declaring complex structures in function definitions so they can be extracted first. Other, simpler kinds of modification may include changing the casing of identifier to adhere to target language coding standards, e.g., camelCase in JavaScript vs snake_case in Python3.

**[0061]** One or more code characteristics can then be propagated **720** from the modified coding node to the intermediate, transformed coding node. The two primary examples for propagation of node characteristic include asynchronous processing and exception handling. Many modern programming languages distinct synchronous and asynchronous modes of processing with a special syntax. If a user calls a asynchronous function they need to do it in a special way. If this happens in another function it makes it an asynchronous function by propagation. The JSX itself does not have the notion of async/sync calls. It uses propagation to figure out which Function and Call nodes should be transpiled to asynchronous syntax in the target programming language. The same applies to exception handling. Some

languages explicitly specify which function can throw errors and this characteristic also propagates across function calls.

**[0062]** The present disclosure, in various aspects, embodiments, and/or configurations, includes components, methods, processes, systems, and/or apparatus substantially as depicted and described herein, including various aspects, embodiments, configurations embodiments, sub-combinations, and/or subsets thereof. Those of skill in the art will understand how to make and use the disclosed aspects, embodiments, and/or configurations after understanding the present disclosure. The present disclosure, in various aspects, embodiments, and/or configurations, includes providing devices and processes in the absence of items not depicted and/or described herein or in various aspects, embodiments, and/or configurations hereof, including in the absence of such items as may have been used in previous devices or processes, e.g., for improving performance, achieving ease and\or reducing cost of implementation.

**[0063]** The foregoing discussion has been presented for purposes of illustration and description. The foregoing is not intended to limit the disclosure to the form or forms disclosed herein. In the foregoing Detailed Description for example, various features of the disclosure are grouped together in one or more aspects, embodiments, and/or configurations for the purpose of streamlining the disclosure. The features of the aspects, embodiments, and/or configurations of the disclosure may be combined in alternate aspects, embodiments, and/or configurations other than those discussed above. This method of disclosure is not to be interpreted as reflecting an intention that the claims require more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed aspect, embodiment, and/or configuration. Thus, the following claims are hereby incorporated into this Detailed Description, with each claim standing on its own as a separate preferred embodiment of the disclosure.

**[0064]** Moreover, though the description has included description of one or more aspects, embodiments, and/or configurations and certain variations and modifications, other variations, combinations, and modifications are within the scope of the disclosure, e.g., as may be within the skill and knowledge of those in the art, after understanding the present disclosure. It is intended to obtain rights which include alternative aspects, embodiments, and/or configurations to the extent permitted, including alternate, interchangeable and/or equivalent structures, functions, ranges or steps to those claimed, whether or not such alternate, interchangeable and/or equivalent structures, functions, ranges or steps are disclosed herein, and without intending to publicly dedicate any patentable subject matter.

What is claimed is:

**1**. A method for automated generation of computer source code, the method comprising:

receiving, by a processor of a code generation system, information defining a coding node, the coding node comprising a definition of an object within an object tree for the computer source code;

creating, by the processor of the code generation system, the coding node based on the received information defining the coding node;

transforming, by the processor of the code generation system, the coding node into an intermediate, transformed coding node; and

transpiling, by the processor of the code generation system, the intermediate, transformed coding node into software source code in a target programming language of a plurality of programming languages.

**2**. The method of claim **1**, wherein the information defining the coding node comprises Javascript Syntax extension (JSX) code.

**3**. The method of claim **2**, wherein the information defining the coding node further comprises user defined information.

**4**. The method of claim **1**, wherein creating the coding node comprises generating one or more of: a node type; one or more node attributes; or one or more child nodes for the coding node based on the received information defining the coding node.

**5**. The method of claim **4**, wherein creating the coding node further comprises performing a preliminary validation on the coding node.

**6**. The method of claim **1**, wherein transforming the coding node into the intermediate, transformed coding node further comprises:

gathering additional information defining a context for the coding node;

validating the coding node for unresolved references and anomalies;

modifying the coding node based on the information defining the coding node and the object tree; and

propagating one or more code characteristics from the modified coding node to the intermediate, transformed coding node.

**7**. The method of claim **1**, wherein transpiling the intermediate, transformed coding node into the source code in the target programming language comprises selecting a transpiler from a plurality of available transpilers based on the target programming language.

**8**. A system comprising:

a processor; and

a memory coupled with and readable by the processor and storing therein a set of instructions which, when executed by the processor, causes the processor to:

receive information defining a coding node, the coding node comprising a definition of an object within an object tree for the computer source code;

create the coding node based on the received information defining the coding node;

transform the coding node into an intermediate, transformed coding node; and

transpile the intermediate, transformed coding node into software source code in a target programming language of a plurality of programming languages.

**9**. The system of claim **8**, wherein the information defining the coding node comprises Javascript Syntax extension (JSX) code.

**10**. The system of claim **9**, wherein the information defining the coding node further comprises user defined information.

**11**. The system of claim **8**, wherein creating the coding node comprises generating one or more of: a node type; one or more node attributes; or one or more child nodes for the coding node based on the received information defining the coding node.

**12**. The system of claim **11**, wherein creating the coding node further comprises performing a preliminary validation on the coding node.

**13**. The system of claim **8**, wherein transforming the coding node into the intermediate, transformed coding node further comprises:

gathering additional information defining a context for the coding node;

validating the coding node for unresolved references and anomalies;

modifying the coding node based on the information defining the coding node and the object tree; and

propagating one or more code characteristics from the modified coding node to the intermediate, transformed coding node.

**14**. The system of claim **8**, wherein transpiling the intermediate, transformed coding node into the source code in the target programming language comprises selecting a transpiler from a plurality of available transpilers based on the target programming language.

**15**. A non-transitory, computer-readable medium comprising a set of instructions stored therein which, when executed by a processor, causes the processor to:

receive information defining a coding node, the coding node comprising a definition of an object within an object tree for the computer source code;

create the coding node based on the received information defining the coding node;

transform the coding node into an intermediate, transformed coding node; and

transpile the intermediate, transformed coding node into software source code in a target programming language of a plurality of programming languages.

**16**. The non-transitory, computer-readable medium of claim **15**, wherein the information defining the coding node comprises Javascript Syntax extension (JSX) code.

**17**. The non-transitory, computer-readable medium of claim **16**, wherein the information defining the coding node further comprises user defined information.

**18**. The non-transitory, computer-readable medium of claim **15**, wherein creating the coding node comprises generating one or more of: a node type; one or more node attributes; or one or more child nodes for the coding node based on the received information defining the coding node.

**19**. The non-transitory, computer-readable medium of claim **18**, wherein creating the coding node further comprises performing a preliminary validation on the coding node.

**20**. The non-transitory, computer-readable medium of claim **15**, wherein transforming the coding node into the intermediate, transformed coding node further comprises:

gathering additional information defining a context for the coding node;

validating the coding node for unresolved references and anomalies;

modifying the coding node based on the information defining the coding node and the object tree; and

propagating one or more code characteristics from the modified coding node to the intermediate, transformed coding node.

* * * * *