

classes **215** (step **332**). Usage check of the script code section is performed by extracting the language elements from the script code section (step **331**), and comparing the extracted language elements with the definition module (step **333**). In order to perform semantics check of the generated script code section, language-dependent code is generated in a language that supports classes and interfaces (step **335**). The language-dependent representation for the script code section generated in step **335** implements classes and associated interfaces. Semantics check of the script code section is performed by compiling the generated interfaces **210** and the classes **215** (step **337**).

[**0025**] FIG. 4 illustrates the validation of a computer program having a compiler-language section and a script code section. The XML implementation modules **100** and definition modules **105** represent a computer program. A syntax checker **107** can be used to perform a syntax check at the XML level for the implementation modules **100** and the definition modules **105**. Program implementation code for the script code section of the computer program **120** is generated from the implementation modules **100** and the definition modules **105** using an XSL processor **110** in conjunction with XSL style sheets **115**. The program implementation code **120** includes a script code implementation of the computer program **121** in a scripting language, e.g., JavaScript or Perl. The program implementation code **120** also includes an implementation of the script code section in an intermediate compiler-language **122** that supports classes and interfaces. A compiler **125** is used to perform implementation checks on the compiler-language implementation of the script code section **122**. If the script language generator works properly, the implementation checks performed by the compiler **125** can be used to conclude that the generated script language implementation **121** works accordingly. Usage checks are performed on the script language implementation **121** using a script language parser **127**. The script language parser **127** performs usage checks by extracting language elements from the generated script language implementation **121** and comparing the language elements with the definition modules **105**.

[**0026**] Program implementation code for the compiler-language portion of the computer program **130** is generated from the implementation modules **100** and the definition modules **105** using the XSL processor **110** in conjunction with XSL style sheets **116**. The compiler **125** verifies the class and interface definitions by performing usage and implementation checks.

[**0027**] FIG. 5 illustrates use of definition and implementation modules to validate a program and generate HTML code. In FIG. 5, a computer program is represented using a script code section **510** and two XML tree structures **500**, **505**. The first XML tree structure **500** describes the implementation modules in the form of classes for an HTML document. The second XML tree structure **505** describes definition modules in the form of associated interfaces. The script code section provides the functionality for the HTML document, e.g., support for the scrolling of text. The script code section can be implemented in any scripting language, e.g., JavaScript or Perl. The syntax checker **507** is used to perform a syntax check at the XML level for the implementation modules **500** and the definition modules **505**. The HTML document **520** is generated from the implementation

modules **500** and the definition modules **505** using the XSL processor **510** in conjunction with XSL style sheets **515**. The script language parser **527** performs usage checks by extracting language elements from the script code section **510** and comparing the language elements with the definition modules **505**.

[**0028**] The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The invention can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

[**0029**] Method steps of the invention can be performed by one or more programmable processors executing a computer program to perform functions of the invention by operating on input data and generating output. Method steps can also be performed by, and apparatus of the invention can be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

[**0030**] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in special purpose logic circuitry.

[**0031**] The invention has been described in terms of particular embodiments. Other embodiments are within the scope of the following claims. For example, the steps of the invention can be performed in a different order and still achieve desirable results.