



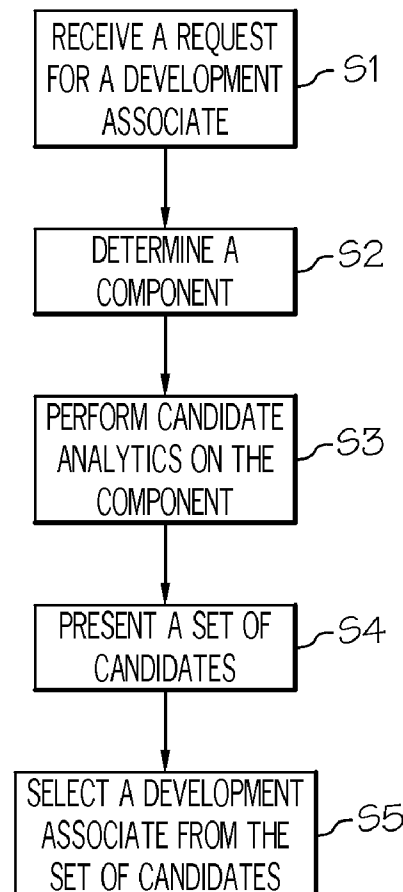
US 20140019933A1

(19) **United States**(12) **Patent Application Publication**
Balasubramanian et al.(10) **Pub. No.: US 2014/0019933 A1**(43) **Pub. Date: Jan. 16, 2014**(54) **SELECTING A DEVELOPMENT ASSOCIATE
FOR WORK IN A UNIFIED MODELING
LANGUAGE (UML) ENVIRONMENT****Publication Classification**(51) **Int. Cl.**
G06F 9/44 (2006.01)(75) Inventors: **Swaminathan Balasubramanian**, Troy,
MI (US); **Rick A. Hamilton, II**,
Charlottesville, VA (US); **Brian M.**
O'Connell, RTP, NC (US); **Harm**
Sluiman, Scarborough (CA)(52) **U.S. Cl.**
USPC **717/102**(73) Assignee: **INTERNATIONAL BUSINESS
MACHINES CORPORATION**,
Armonk, NY (US)(57) **ABSTRACT**

An approach is provided for selecting a development resource to develop a component in a Unified Modeling Language (UML) software system architecture. Specifically, a request for a developer is received from an architect. A component for the developer is determined and candidate analytics is performed on the component. Based on the candidate analytics, a list of potential candidates is presented to the architect. A developer is then selected from the list of candidates.

(21) Appl. No.: **13/546,301**(22) Filed: **Jul. 11, 2012**

300



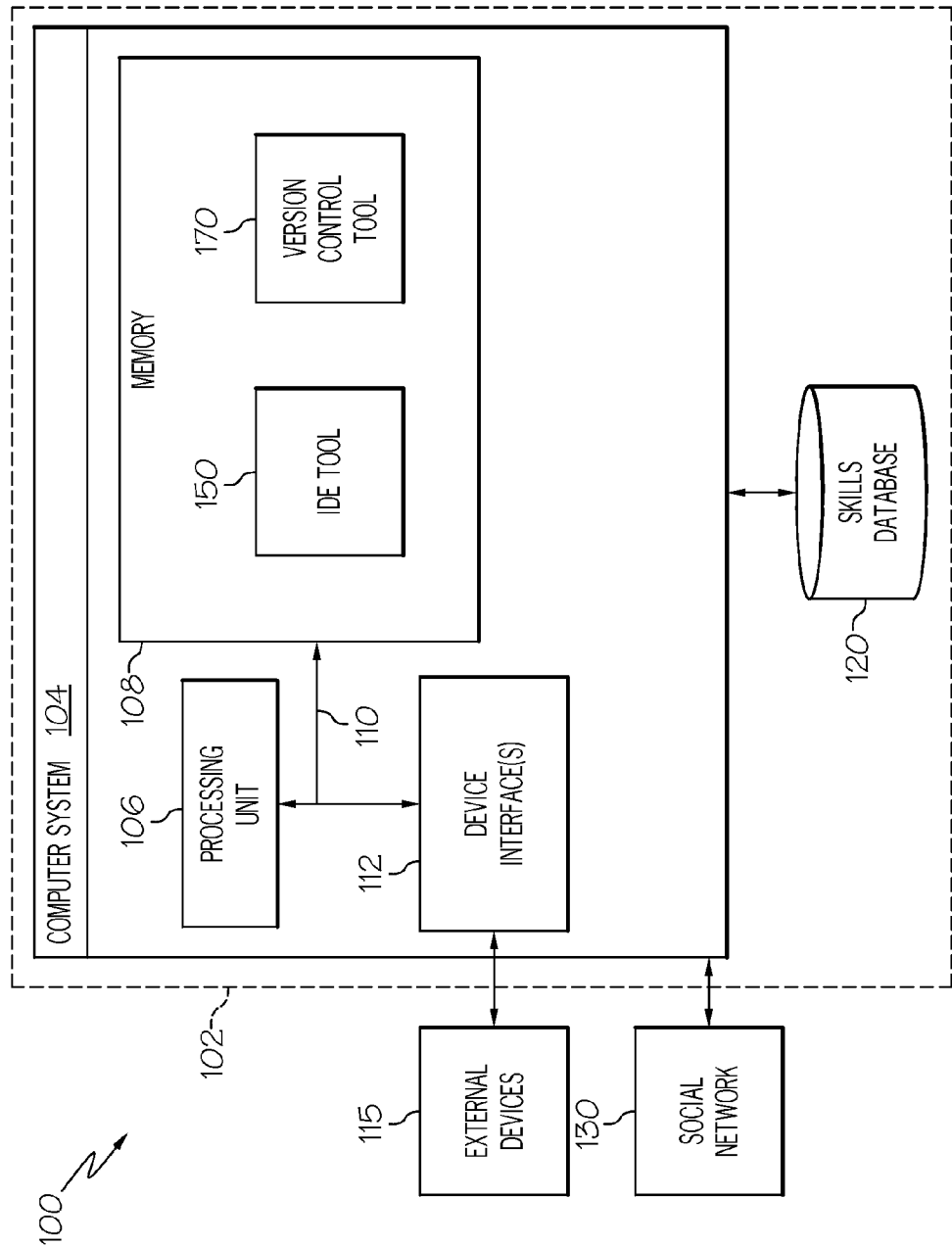


FIG. 1

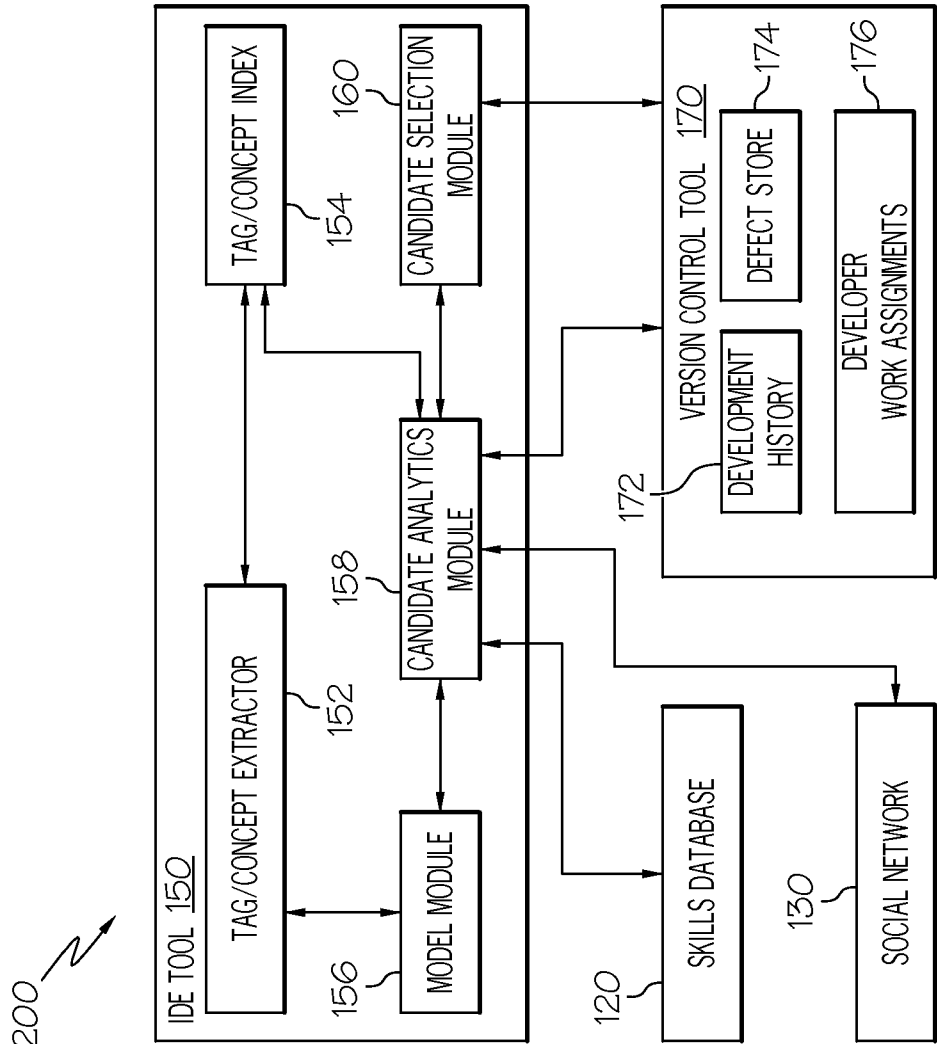


FIG. 2

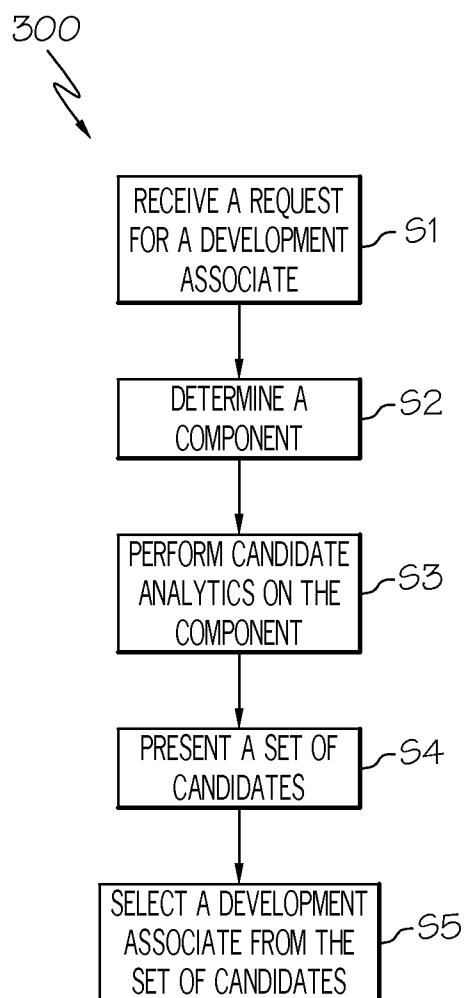
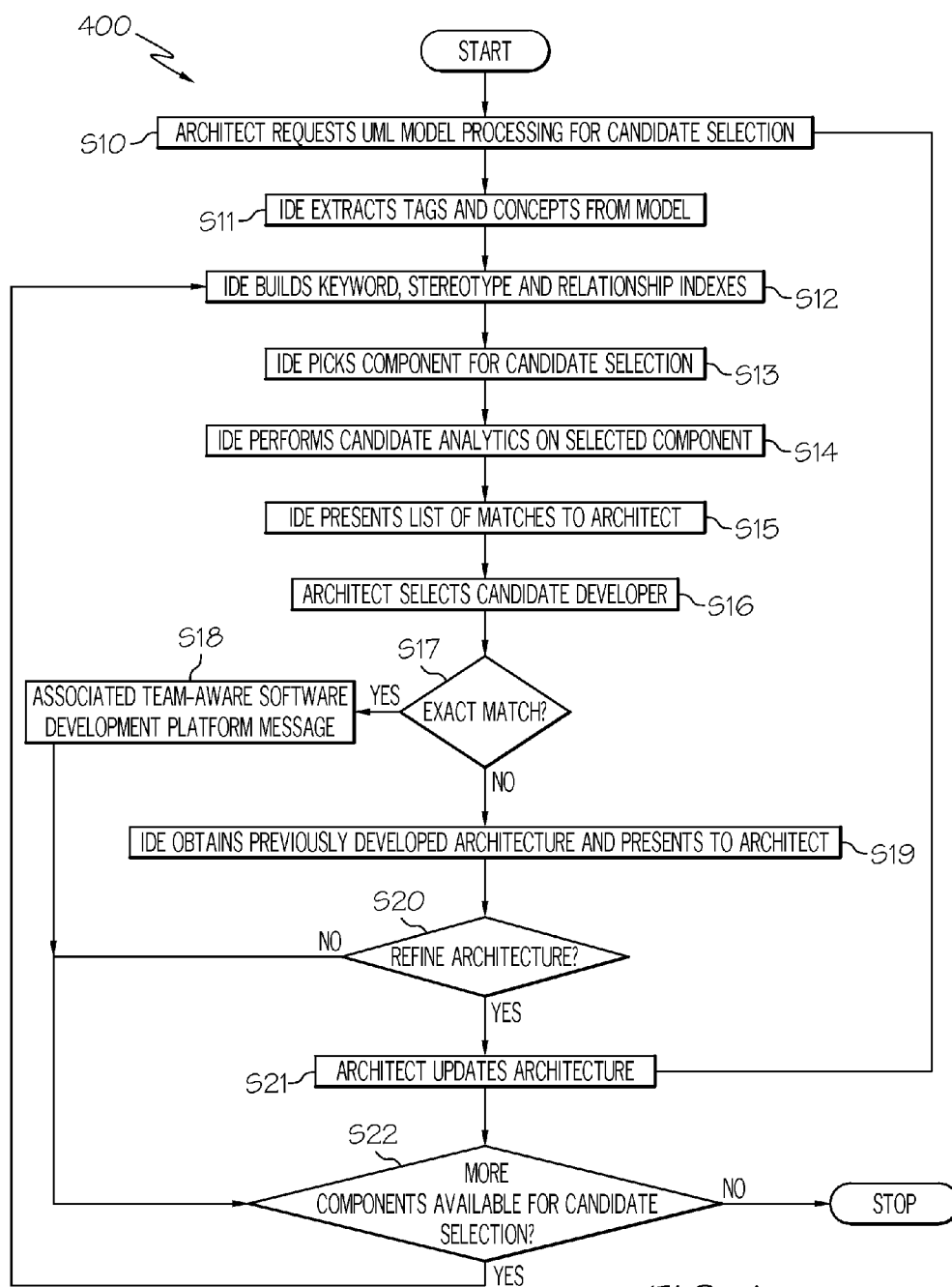


FIG. 3



SELECTING A DEVELOPMENT ASSOCIATE FOR WORK IN A UNIFIED MODELING LANGUAGE (UML) ENVIRONMENT

TECHNICAL FIELD

[0001] This invention relates generally to a software development environment and, more specifically, to selecting a development associate to develop a component in a Unified Modeling Language (UML) software system architecture.

BACKGROUND

[0002] Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts or components of a software-intensive system. UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions, as well as concrete things such as programming language statements, database schemas, and reusable software components. UML combines techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes throughout the software development life cycle and across different implementation technologies. UML has synthesized the notations of the Booch method, the object-modeling technique (OMT), and object-oriented software engineering (DOSE) by fusing them into a single common and widely usable modeling language.

[0003] Many large corporations have geographically dispersed technical development teams. Due to the dispersed nature of these teams and changing resources in work environments, it can be difficult for software architects to know which development associate to assign to a newly architected software component that will be used in building a system.

SUMMARY

[0004] In general, an approach is provided for selecting a development resource to develop a component in a Unified Modeling Language (UML) software system architecture. Specifically, a request for a developer is received from an architect. A component for the developer is determined and candidate analytics is performed on the component. Based on the candidate analytics, a list of potential candidates is presented to the architect. A developer is then selected from the list of candidates.

[0005] A first aspect of the present invention includes a computer-implemented method for selecting a development associate in a Unified Modeling Language (UML) software system architecture, comprising: determining a component for the development associate; performing candidate analytics on the component; presenting a set of candidates based on the candidate analytics; and selecting a development associate from the set of candidates.

[0006] A second aspect of the present invention provides a system for selecting a development associate in a Unified Modeling Language (UML) software system architecture, comprising: a memory medium comprising instructions; a bus coupled to the memory medium; and a processor coupled to an integrated development environment (IDE) tool via the bus that when executing the instructions causes the system to: determine a component for the development associate; perform candidate analytics on the component; present a set of

candidates based on the candidate analytics; and select a development associate from the set of candidates.

[0007] A third aspect of the present invention provides a computer program product for selecting a development associate in a Unified Modeling Language (UML) software system architecture, the computer program product comprising a computer readable storage medium, and program instructions stored on the computer readable storage medium, to: determine a component for the development associate; perform candidate analytics on the component; present a set of candidates based on the candidate analytics; and select a development associate from the set of candidates.

[0008] A fourth aspect of the present invention includes a method for providing a system for selecting a development associate in a Unified Modeling Language (UML) software system architecture, comprising: determining a component for the development associate; performing candidate analytics on the component; presenting a set of candidates based on the candidate analytics; and selecting a development associate from the set of candidates.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] These and other features of this invention will be more readily understood from the following detailed description of the various aspects of the invention taken in conjunction with the accompanying drawings in which:

[0010] FIG. 1 shows a schematic of an exemplary computing environment according to illustrative embodiments of the present invention.

[0011] FIG. 2 shows a pictorial representation of a system in which aspects of the illustrative embodiments of the present invention may be implemented.

[0012] FIG. 3 shows a process flow diagram for selecting a development associate in a Unified Modeling Language (UML) software system architecture according to illustrative embodiments of the present invention.

[0013] FIG. 4 shows a more detailed process flow diagram for selecting a development associate in a Unified Modeling Language (UML) software system architecture according to illustrative embodiments of the present invention.

[0014] The drawings are not necessarily to scale. The drawings are merely representations, not intended to portray specific parameters of the invention. The drawings are intended to depict only typical embodiments of the invention, and therefore should not be considered as limiting in scope. In the drawings, like numbering represents like elements.

DETAILED DESCRIPTION

[0015] Exemplary embodiments will now be described more fully herein with reference to the accompanying drawings, in which exemplary embodiments are shown. An approach is provided for selecting a development resource to develop a component in a Unified Modeling Language (UML) software system architecture. Specifically, a request for a developer is received from an architect. A component for the developer is determined and candidate analytics is performed on the component. Based on the candidate analytics, a list of potential candidates is presented to the architect. A developer is then selected from the list of candidates.

[0016] It will be appreciated that this disclosure may be embodied in many different forms and should not be construed as limited to the exemplary embodiments set forth herein. Rather, these exemplary embodiments are provided so

that this disclosure will be thorough and complete and will fully convey the scope of this disclosure to those skilled in the art. The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of this disclosure. For example, as used herein, the singular forms “a”, “an”, and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. Furthermore, the use of the terms “a”, “an”, etc., do not denote a limitation of quantity, but rather denote the presence of at least one of the referenced items. It will be further understood that the terms “comprises” and/or “comprising”, or “includes” and/or “including”, when used in this specification, specify the presence of stated features, regions, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, regions, integers, steps, operations, elements, components, and/or groups thereof.

[0017] Reference throughout this specification to “one embodiment,” “an embodiment,” “embodiments,” or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases “in one embodiment,” “in an embodiment,” “in embodiments” and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

[0018] FIG. 1 illustrates a computerized implementation 100 of the present invention. As depicted, implementation 100 includes computer system 104 deployed within a computer infrastructure 102. This is intended to demonstrate, among other things, that the present invention could be implemented within a network environment (e.g., the Internet, a wide area network (WAN), a local area network (LAN), a virtual private network (VPN), etc.), or on a stand-alone computer system. In the case of the former, communication throughout the network can occur via any combination of various types of communications links. For example, the communication links can comprise addressable connections that may utilize any combination of wired and/or wireless transmission methods. Where communications occur via the Internet, connectivity could be provided by conventional TCP/IP sockets-based protocol, and an Internet service provider could be used to establish connectivity to the Internet. Still yet, computer infrastructure 102 is intended to demonstrate that some or all of the components of implementation 100 could be deployed, managed, serviced, etc., by a service provider who offers to implement, deploy, and/or perform the functions of the present invention for others.

[0019] Computer system 104 is intended to represent any type of computer system that may be implemented in deploying/realizing the teachings recited herein. In this particular example, computer system 104 represents an illustrative system for selecting a development associate in a Unified Modeling Language (UML) software system architecture. It should be understood that any other computers implemented under the present invention may have different components/software, but will perform similar functions. As shown, computer system 104 includes a processing unit 106, memory 108 for storing an IDE tool 150 and/or version control tool 170, a bus 110, and device interfaces 112.

[0020] Processing unit 106 collects and routes signals representing outputs from external devices 115 (e.g., a keyboard, a pointing device, a display, a graphical user interface, etc.) to IDE tool 150 and/or version control tool 170. The signals can

be transmitted over a LAN and/or a WAN (e.g., T1, T3, 56 kb, X.25), broadband connections (ISDN, Frame Relay, ATM), wireless links (802.11, Bluetooth, etc.), and so on. In some embodiments, the signals may be encrypted using, for example, trusted key-pair encryption. Different external devices may transmit information using different communication pathways, such as Ethernet or wireless networks, direct serial or parallel connections, USB, Firewire®, Bluetooth®, or other proprietary interfaces. (Firewire is a registered trademark of Apple Computer, Inc. Bluetooth is a registered trademark of Bluetooth Special Interest Group (SIG)).

[0021] In general, processing unit 106 executes computer program code, such as program code for IDE tool 150 and/or version control tool 170, which is stored in memory 108 and/or skills database 120. While executing computer program code, processing unit 106 can read and/or write data to/from memory 108 and/or skills database 120. Skills database 120 can include VCRs, DVRs, RAID arrays, USB hard drives, optical disk recorders, flash storage devices, or any other similar storage device. Although not shown, computer system 104 could also include I/O interfaces that communicate with one or more external devices 115 that enable a user to interact with computer system 104.

[0022] FIG. 2 shows a more detailed view of an example system diagram 200 for implementing methods according to an embodiment of the present invention. Example system diagram 200 consists of IDE tool 150, version control tool 170, skills database 120, and social network 130. IDE tool 150 may include tag/concept extractor 152, tag/concept index 154, model module 156, candidate analytics module 158, and candidate selection module 160. Candidate analytics module 158 may communicate with skills database 120 and/or social network 130. Version control tool 170 may be configured to read and/or write development history 172, defect store 174, and developer work assignments 176.

[0023] Within IDE tool 150, in a typical embodiment, model module 156 may use tag/concept extractor 152 to process a supplied UML model and build a tag/concept index 154. Candidate analytics module 158 may use the tag/concept index 154 along with version control tool 170 (consisting of development history 172, defect store 174, and developer work assignments 176), skills database 120, and social network 130 to build a list of resource candidates. Finally, the candidate selection module 160 may present exact and partial matches to the architect using historic architecture data from version control tool 170. A development associate (e.g., software developer, information developer, project manager, technical writer, and the like) is then selected. In one example, the development associate is selected during a design phase of a project.

[0024] UML allows the architecture model of a system being developed to be specified at various levels of detail. Commonly developed models include system context, architecture overview, analysis (or platform independent) design, and enterprise (or platform specific) design. In one example, an architect may develop a UML model consisting of, but not limited to, components, component associations (structural view), component interactions (behavioral view), and component deployment (operational view). A component in the UML represents a modular part of a system that encapsulates its content and whose manifestation is replaceable within its environment. A component defines its behavior in terms of

provided and required interfaces. A component may be replaced by another if, and only if, their provided and required interfaces are identical.

[0025] UML enables the assignment of snippets of information to various elements of the architectural model. A UML profile provides a means of extending UML, which enables the architect to build UML models in particular domains. The profile is based on additional stereotypes and tagged values that are applied to elements, attributes, methods, connectors, connector ends, and so on. The assignment of the snippets of information may use stereotypes and stereotype properties via the application of one or more UML profiles. Alternatively, UML keywords or comments may be used. Using either approach, the architect may assign one or more keywords (or tags) and/or assign requirements to, for example, an element.

[0026] FIG. 3 shows a process flow diagram 300 for selecting a development associate in a UML software system architecture according to illustrative embodiments of the present invention. At S1, a request for a development associate (or technical resource) is received. At S2, a component for the development associate is determined. At S3, candidate analytics on the component are performed. At S4, a set (one or more) of candidates are presented based on the candidate analytics. Finally, at S5, a development associate is selected from the set of candidates that are presented. Table 1 (shown below) is used to describe an example selection process according to embodiments of the present invention.

TABLE 1

Component: User Authentication Module	
Stereotypes:	Authentication (properties: client:java; auth-method:webservices; security:https); Prog-lang (properties: lang:java; version:1.5)
Keywords:	corporate_directory; web services, sso, singesignon, wsmr
Comments:	REQ01.01 - The system shall authenticate the user against the corporate directory; REQ01.02 - The system shall not require re-authentication if the user has already been authenticated previously.

[0027] In Table 1 above, the architect is designing an authentication module called 'User Authentication Module'. A corporate UML profile is assigned and a stereotype called 'authentication' is selected. Values are assigned to the selected stereotype's properties indicating that the authentication is being called from a Java® program using web services over hypertext transfer protocol secure (https) (Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates). Also, a stereotype called 'prog-lang' (i.e., programming language) is selected indicating the module needs to be developed using Java® (version 1.5). In addition, a set of keywords is assigned indicating that the module will use the corporate directory for authentication over web services using single signon (SSO) making use of a service repository. Finally, in this example, two requirements (REQ01.01 and REQ01.02) are assigned that the authentication module component must fulfill. In other examples, any number of stereotypes, keywords, and/or requirements may be associated with a respective component.

[0028] FIG. 4 shows a more detailed process flow 400 for selecting a development associate during UML model processing according to illustrative embodiments of the present invention. At S10, an architect requests UML model processing for selecting a candidate. At S11, the tag/concept extractor

152 of the IDE tool 150 extracts tags and/or concepts. Tag/concept extractor 152 may be configured to extract tags and concepts from elements of the architecture model. At S12, the IDE tool 150 builds keyword, stereotype and/or relationship indexes. The keyword index may hold the keyword(s) assigned to each UML element. Known methods may be employed to extract concepts from the requirements associated with the respective element. In one example, these concepts may also be placed into a keyword index. Since stereotypes are more structured with semantics associated with them, one or more stereotype indexes may be built to hold their properties. In one example, a stereotype index is built for each stereotype. UML naturally supports relationships between elements which are both implicit and explicit. A relationship index may be built to store such relationships to enable roll-up of tags/concepts.

[0029] At S13, the IDE tool 150 determines a component for candidate selection. At S14, the candidate analytics module 158 performs candidate analytics on the selected component. The candidate analytics module 158 may be configured to enable the architect to select the best developer resource for each newly designed component using a plurality of sources including, but not limited to, skills database 120, version control tool 170, and social network 130. Social network 130 may include the architect's social network and peer architects' social networks.

[0030] Development history 172 may include each developer's component development history. The development history may be recorded using known techniques. The type of each component may be identified automatically through the UML design that is associated with the component creation. For example, the UML design may specify the component interacts with a database, a network socket, SOAP, or the like. Each of these may be considered "tags" or component keywords. In one example, more than one keyword may be assigned to each component. Each developer may be linked to their component history and each keyword may be linked to one or more developers. In one example, the information may be kept in a relational data store to enable rapid searching and sorting. Individuals with more experience with a component type may be preferred over other individuals when selecting optimal resource candidates.

[0031] Each defect against any component or source files is stored. In one example, defect information may be stored in version control tool 170. The author of each component may already be stored in development history 172 as described above. In one example, a defect count is increased for the responsible developer and the corresponding defect count for the associated keywords for each component. For example, if the component was related to database design, the developer's database design defect count would be increased. In some examples, the defect count may be stored as a defect per component count such that there is no advantage for developers with less experience. In one example, individuals with fewer defects per component type may be preferred over other individuals when selecting optimal resource candidates.

[0032] The workload for each developer may be stored in developer work assignments 176. Each new assignment is stored and the responsible developer's workload count may be increased upon assignment of a new task. Upon assignment completion, the responsible developer's workload count may be decreased. The workload count represents the total number of outstanding tasks for each developer. In one

example, individuals with a lower workload may be preferred over other individuals when selecting optimal resource candidates.

[0033] Other information may be used when nominating one or more optimal resource candidates to be considered for a particular assignment, such as community activity, crowd-sourcing, scanned documents such as curriculum vitae (CV's) or resumes having predefined keywords, scanned skills databases, and the like. Community activity may include the social interactions of developers. Social interactions may include activities such as article or journal authoring of developers, activities on various developer forums, and the like.

[0034] Crowd-sourcing involves parties having knowledge of skills nominating or voting on those individuals with the best ability to undertake a particular assignment. Such nomination may take place either in binary form (e.g., statement as to whether a particular person is well suited for the role), relative form (e.g., Sue is best, then Sanjay, then Robert), or through subjective skills assessment e.g., Sue's skills are 5/5, Sanjay's are 4/5, and Robert's are 4/5). These inputs may be aggregated to either automatically guide role placement or to provide decision support for key personnel, such as project managers or resource deployment managers.

[0035] One or more scanned documents (e.g., CV, resume, or the like) having one or more predefined keywords may also be used. The documents may be scanned to assess an individual's acceptability for a given role. In this method, an individual having a resume containing more keywords than another individual may be considered a stronger candidate for the role. Alternatively, or in addition, an individual candidate demonstrating multiple projects with a given skill than another individual may be considered a stronger match. Also, an individual showing a longer length of time using a certain skill than another individual may be considered a stronger candidate for the role.

[0036] One or more skills databases may be scanned when nominating one or more optimal resource candidates to be considered for a particular assignment. A skills database may contain employee skills sets. These skill sets may be self-graded (i.e., an individual herself issues as assessment on her strengths and weaknesses) or developed by others (e.g., peers or management staff). Again, similar to the technique involving scanned documents, individuals having more keywords, higher rankings, greater time, and/or higher number of past roles may be considered stronger candidates for the role.

[0037] Referring back to FIG. 4, the IDE tool **150** presents a list of candidate matches to the architect at **S15**. When the architect requests a candidate selection, the candidate analytics module **158** may be invoked, and a list of candidates is built using one or more of the techniques described above. In one example, the candidate analytics module **158** may classify each selected candidate resource as either an exact match or a partial match.

[0038] In one example, the candidate analytics module **158** may classify a given resource as an exact match when the development associate has performed or successfully fulfilled the role on a previous occasion. In other examples, a development associate may be classified as an exact match when meeting other predefined criteria. The architect may select an individual from the exact match list for assignment to the component. Upon selection, the new component may be created and the development associate assignment may be made.

[0039] A partial match may be best described using an example. Consider the example in which the architect has a component for authentication using Simple Object Access Protocol (SOAP) application programming interface (API) deployed on a WebSphere Application Server. The candidate analytics module **158** finds a developer who has previously implemented authentication using Lightweight Directory Access Protocol (LDAP). In this example, the developer resource is treated as a partial match. In the case of a partial match, in one example, the method obtains the architecture previously developed associated with the developer from a repository, such as Rational Team Concert™ (trademark of IBM Corp.), presents the previously developed architecture to the architect, and recommends refinement of the architecture model. If the architect proceeds to refine the proposed architecture to be in line with the developer's experience, candidate selection may be re-executed on the updated architecture model, thus enabling the partial match to become an exact match.

[0040] At **S16**, the architect selects a development associate (such as a software developer) from the list of candidate matches. At **S17**, a determination is made whether the selected developer is an exact match. If an exact match, an associated team-aware software development platform message is sent at **S18**. If not an exact match, the IDE tool **150** obtains a previously developed architecture and presents it to the architect at **S19**. At **S20**, a determination is made whether to refine the architecture. If the architecture is to be refined, the architect updates the architecture at **S21**. At **S22**, a determination is made whether additional components are available for candidate selection. If additional components are available, then the process is repeated.

[0041] Further, it can be appreciated that the methodologies disclosed herein can be used within a computer system to provide a developer resource selection service, as shown in FIG. 1. In this case, IDE tool **150** and version control tool **170** can be provided, and one or more systems for performing the processes described in the invention can be obtained and deployed to computer infrastructure **102**. To this extent, the deployment can comprise one or more of (1) installing program code on a computing device, such as a computer system, from a computer-readable medium; (2) adding one or more computing devices to the infrastructure; and (3) incorporating and/or modifying one or more existing systems of the infrastructure to enable the infrastructure to perform the process actions of the invention.

[0042] The exemplary computer system **104** may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, people, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types. Exemplary computer system **104** may be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage medium including memory storage devices.

[0043] Furthermore, an implementation of exemplary computer system **104** may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a com-

puter. By way of example, and not limitation, computer readable media may comprise “computer storage medium” and “communications media.”

[0044] “Computer storage medium” includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage medium includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer.

[0045] “Communication media” typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier wave or other transport mechanism. Communication media also includes any information delivery media.

[0046] 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 wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above are also included within the scope of computer readable media.

[0047] It is apparent that there has been provided with this invention an approach for selecting a development associate to develop a component in a Unified Modeling Language (UML) software system architecture. While the invention has been particularly shown and described in conjunction with a preferred embodiment thereof, it will be appreciated that variations and modifications will occur to those skilled in the art. Therefore, it is to be understood that the appended claims are intended to cover all such modifications and changes that fall within the true spirit of the invention.

What is claimed is:

1. A computer-implemented method for selecting a development associate in a Unified Modeling Language (UML) software system architecture, comprising:

- determining a component for the development associate;
- performing candidate analytics on the component;
- presenting a set of candidates based on the candidate analytics; and
- selecting a development associate from the set of candidates.

2. The computer-implemented method of claim 1, wherein the development associate is one of a software developer, information developer, project manager, or technical writer.

3. The computer-implemented method of claim 1, wherein the development associate is selected during a design phase of a project.

4. The computer-implemented method of claim 1, wherein the step of performing candidate analytics is based on at least one of a resource skills database or a social network.

5. The computer-implemented method of claim 1, further comprising classifying each of the set of candidates.

6. The computer-implemented method of claim 5, wherein each of the set of candidates are classified as a match or non-match.

7. The computer-implemented method of claim 1, wherein the step of performing candidate analytics is based on at least one of a development history, a set of defects, or a set of work assignments.

8. A system for selecting a development associate in a Unified Modeling Language (UML) software system architecture, comprising:

- a memory medium comprising instructions;
- a bus coupled to the memory medium; and
- a processor coupled to an integrated development environment (IDE) tool via the bus that when executing the instructions causes the system to:
 - determine a component for the development associate;
 - perform candidate analytics on the component;
 - present a set of candidates based on the candidate analytics; and
 - select a development associate from the set of candidates.

9. The system of claim 7, wherein the development associate is one of a software developer, information developer, project manager, or technical writer.

10. The system of claim 7, wherein the development associate is selected during a design phase of a project.

11. The system of claim 7, wherein the candidate analytics is performed based on at least one of a resource skills database or a social network.

12. The system of claim 7, the instructions causing the system to classify each of the set of candidates.

13. The system of claim 12, wherein each of the set of candidates is classified as a match or non-match.

14. The system of claim 7, wherein the candidate analytics is performed based on at least one of a development history, a set of defects, or a set of work assignments.

15. A computer program product for selecting a development associate in a Unified Modeling Language (UML) software system architecture, the computer program product comprising a computer readable storage medium, and program instructions stored on the computer readable storage medium, to:

- determine a component for the development associate;
- perform candidate analytics on the component;
- present a set of candidates based on the candidate analytics; and
- select a development associate from the set of candidates.

16. The computer program product of claim 15, wherein the development associate is one of a software developer, information developer, project manager, or technical writer.

17. The computer program product of claim 15, wherein the development associate is selected during a design phase of a project.

18. The computer program product of claim 15, wherein the candidate analytics is performed based on at least one of a resource skills database or a social network.

19. The computer program product of claim 15, the computer readable storage medium further comprising instructions to classify each of the set of resources.

20. The computer program product of claim 19, wherein each of the set of candidates is classified as a match or non-match.

21. The computer program product of claim 15, wherein the candidate analytics is performed based on at least one of a resource skills database or a social network.

22. A computer-implemented method for deploying a system for selecting a development associate in a Unified Modeling Language (UML) software system architecture, comprising:

- providing a computer infrastructure being operable to:
 - determine a component for the development associate;
 - perform candidate analytics on the component;
 - present a set of candidates based on the candidate analytics; and
 - select a development associate from the set of candidates.

* * * * *