(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/25909 A2

(51) International Patent Classification[7]: G06F 9/44

(21) International Application Number: PCT/US00/27123

(22) International Filing Date: 2 October 2000 (02.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/156,962 1 October 1999 (01.10.1999) US

(71) Applicant *(for all designated States except US)*: **ANDERSEN CONSULTING L.L.P.** [US/US]; 100 South Wacker Drive, Chicago, IL 60603 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **ARVANITIS, Yannis, S.** [US/US]; 532 W. 4th Street, Hinsdale, IL 60521 (US). **MESOY, Tor** [NO/NO]; Storengvn. 63A, N-1368 Stabekk (NO).

(74) Agent: **MCCONNELL, Dean, E.**; Brinks Hofer Gilson & Lione, One Indiana Square, Suite 2425, Indianapolis, IN 46204 (US).

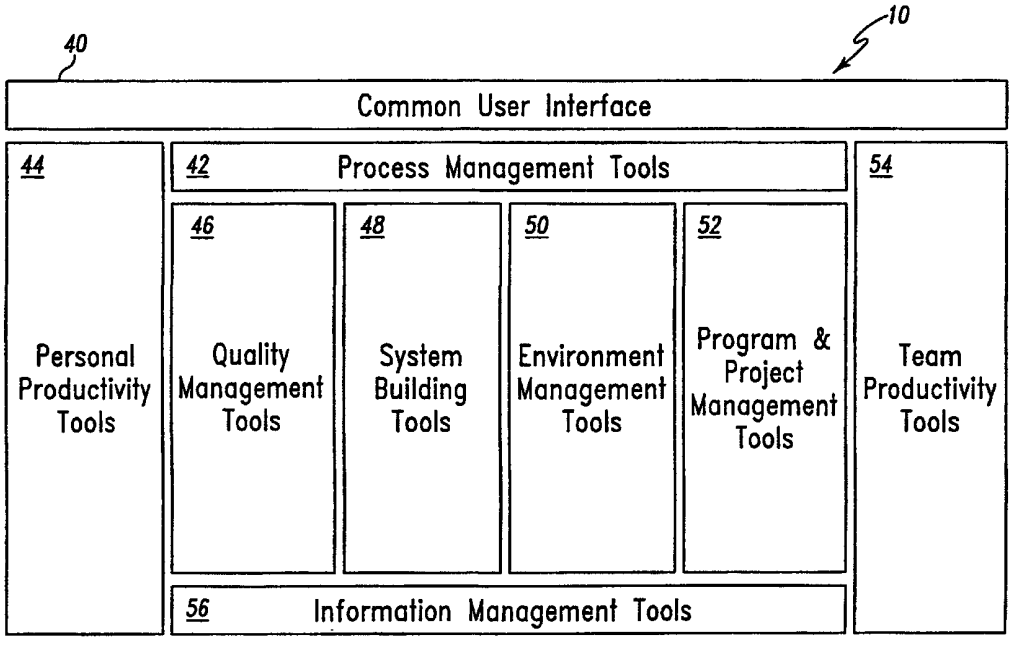
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

[Continued on next page]

(54) Title: DEVELOPMENT ARCHITECTURES FOR NETCENTRIC COMPUTING SYSTEMS



(57) Abstract: A development architecture for a netcentric computing system that includes at least one server connected with a client. The server provides a common user interface between the server and the client. In addition, the server also provides at least one process management tool, at least one personal productivity tool, at least one quality management tool, at least one system building tool, at least one environment management tool, at least one program and project management tool, at least one personal productivity tool and at least one information management tool that is used in the development of the netcentric computing system.

WO 01/25909 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DEVELOPMENT ARCHITECTURES FOR NETCENTRIC COMPUTING SYSTEMS

This application claims the benefit under 35 U.S.C. §119(e) of U.S. provisional application Serial No: 60/156,962 filed on October 1, 1999.

Field of the Invention

The present invention relates generally to business computing systems, and more particularly, to development architectures for netcentric computing systems.

Background of the Invention

Computer-based business solutions have existed for various different types of transactions since the mid-to-late 1960s. During this time period, the technology focused on the use of batch technology. In batch processing, the business user would present a file of transactions to the application. The computer system would then run through the transactions, processing each one, essentially without user intervention. The system would provide reporting at some point in the batch processing. Typically, the reports would be batch-printed, which, in turn, would be used by the business user to correct the input transactions that were resubmitted along with the next batch of transactions.

In the 1970s, businesses began a transition to on-line, interactive transactions. At a conceptual level, this processing opened up the file of transactions found in batch transactions and allowed the user to submit them one at a time, receiving either immediate confirmation of the success of the transaction or else feedback on the nature of the transaction error. The conceptually simple change of having the user interact with the computer on a transaction-at-a-time basis caused huge changes in the nature of business computing. More important, users saw huge changes in what they could do on a day-to-day basis. Customers were no longer forced to wait for a batch run to process the particular application. In essence, the computer had an impact on the entire work flow of the business user.

Along with the advent of on-line interactive systems, it was equally significant that the systems provided a means for the business user to communicate with others in the business as the day-to-day business went along. This capability was provided on the backbone of a wide area network (WAN). The WAN was in itself a demanding technology during this time period and, because of these demands, telecommunications groups emerged within organizations, charged with the responsibility to maintain, evolve and manage the network over a period of time.

The theme of the 1980s was database management systems (DBMSs). Organizations used and applied database technology in the 1970s, but in the 1980s, they grew more confident in the application of DBMS technology. Because of the advances in network technology, the focus was now on the sharing of data across organizational and application boundaries. Curiously, database technology did not change the fundamental way in which business processing was done. DBMS made it more convenient to access the data and to ensure that it could be updated while maintaining the integrity of the data.

In the 1990s, technology began to shift toward client/server computing. Client/server computing is a style of computing involving multiple processors, one of which is typically a workstation, and across which a single business transaction is completed. Using the workstation, the transaction entered by the user could now be processed on a keystroke-by-keystroke basis.

Furthermore, there was a change in the communications. With client/server, users could communicate with others in the work group via a local area network (LAN). The LAN permitted workstation-to-workstation communications at speeds of 100 to 1,000 times what was typically available on a WAN. The LAN was a technology that could be grown and evolved in a local office with little need for direct interaction from the telecommunications group.

During the late 1990s, the Internet began to receive widespread use by consumers and businesses. In the business world, the Internet has caused the concept of business users to expand greatly because of the way in which computers are now capable of being interconnected. In addition, the cost of computers has dropped to the point that it is affordable for almost every household to own a computer if so desired. As such, a need to expand the reach of computing both within and outside the enterprise, and that enables the sharing of data and content between individuals and applications has developed.

Summary of the Invention

The present invention discloses a development architecture, and a method of providing a development architecture, for a netcentric computing system. The preferred development architecture includes a server that is connected with a client. The server provides a common user interface between the server and the client, which is preferentially accomplished, at least in part, with the use of a web browser on the client. A personal productivity tool is provided that may be selected from the group consisting of a spreadsheet

application, a graphic application, a word processor application and a personal calendar application for use by said client and said server.

A quality management tool is also provided for assuring that a predetermined agreed upon level of quality is maintained by the netcentric computing system. As such, the quality management tool monitors transactions and the performance of applications used on the netcentric computing system to ensure an adequate level of quality is maintained. A system building tool is also provided for designing, building and testing applications on the netcentric computing system. An environment management tool for monitoring the performance of said netcentric computing system;

The preferred embodiment of the development architecture also includes a program and project management tool for planning, scheduling, tracking and reporting on project segments in the netcentric computing system. A team productivity tool is also included in the preferred embodiment that allows users on the clients to communicate with other users in the netcentric computing system. An information management tool is also provided including a development repository, a folder management tool and a repository management tool. Further, a process management tool is also provided that allows a respective tool to communicate with another respective tool in the netcentric computing system.

In the preferred embodiment of the present invention, the system building tool may be selected from the group consisting of an analysis and design tool, a reverse engineering tool, a construction tool and a configuration management tool. The analysis and design tool may be selected from the group consisting of a data modeling tool, a process modeling tool, a event modeling tool, a database design tool, application logic design tool, a presentation and design tool, a communication design tool, a performance modeling tool and a component modeling tool. The reverse engineering tool may be selected from the group consisting of a system structure analysis tool, an extraction tool, a repository population tool and a restructuring tool. The construction tool of the system building tools may be selected from the group consisting of a publishing and page mark-up tool, a source code editor tool, a generation tool, a compiler/liker/interpreter/debugger tool and a construction utility tool. The configuration management tool includes a version control tool and a migration control tool.

The environment management tool of the preferred development architecture may be selected from the group consisting of a service management tool, a system management tool, a managing change tool and a service planning tool. The program and project management tool of the preferred development architecture may be selected from the group consisting of a planning tool, a scheduling tool, a tracking tool and a reporting tool. The team productivity

tool may be selected from the group consisting of an E-mail tool, a teamware tool, a publishing tool, a group calendar tool and a methodology browsing tool.

The presently disclosed development architecture provides an optimal development environment for a netcentric computing system. As set forth above, the development architecture provides a combination of development tools that are used as part of the preferred development architecture. These tools allow enterprises to maintain operations and develop new applications to be used on the netcentric computing system, thereby allowing the enterprises to constantly make updates and integrate change in the netcentric computing system.

Further objects and advantages of the present invention will be apparent from the following description, reference being made to the accompanying drawings wherein preferred embodiments of the present invention are clearly shown.

Brief Description of the Drawings

Figure 1 illustrates a development architecture for a netcentric computing system.

Figure 2 illustrates a representative netcentric computing system.

Figure 3 depicts the preferred personal productivity tools of the development architecture.

Figure 4 depicts the preferred quality management tools of the development architecture.

Figure 5 depicts the preferred system building tools of the development architecture.

Figure 6 depicts the preferred environment management tools of the development architecture.

Figure 7 depicts the preferred program and project management tools of the development architecture.

Figure 8 depicts the preferred team productivity tools of the development architecture.

Figure 9 depicts the preferred information management tools of the development architecture.

Detailed Description of the Presently Preferred Embodiments of the Invention

Referring to Figs. 1 and 2, the present invention discloses a development architecture 10 that is preferentially part of a technical architecture (not illustrated) used in a netcentric computing system 12. The preferred netcentric computing system 12 includes at least one client 14 that is connected, via a network connection, to at least one server 16, 18, 20. The

development architecture 10 is used by the servers 16, 18, 20 and the clients 14, 34 during operation of the netcentric computing system 12, as set forth in greater detail below. Referring to Fig. 2, the physical picture of an illustrative netcentric computing system 12 is illustrated. In this example, a business enterprise 22 includes at least one client 14, at least one database server 16, at least one firewall 24, at least one application server 18, at least one web server 20 and a local area network (LAN) connection 26, which are electrically connected as illustrated in Fig. 2.

As generally known in the art, LAN connections 26 generally include software applications and various computing devices (i.e. - network cards, cables, hubs, routers, etc.) that are used to interconnect various computing devices (i.e. - clients 14 and servers 16, 18, 20) that are located at a first business enterprise location 28 to form a computing network at that location. The term LAN connection 28, as used herein, should be broadly construed to include any and all hardware devices and software applications that allows clients 14, servers 16, 18, 20 or other computing devices to be electrically connected together so that they can share and transfer data between devices over the network. Although not illustrated, other devices and resources, such as printers for example, may be connected with the LAN connection 26 so that the devices and resources are available to users of the network. Those skilled in the art would recognize that various types of LAN connections 26 exist and may be used in the present invention.

For the purpose of the present invention, the firewall 24 is used to isolate internal systems from unwanted intruders. In particular, firewalls 24 isolate web servers 20 from all Internet traffic that is not relevant to the netcentric computing system 12. In the preferred embodiment, the only requests allowed through the firewall 24 are for services located on the web servers 20. All requests for other applications (e.g., FTP, Telnet) and other IP addresses that the netcentric computing system 12 receives are typically blocked by the firewall 24 during operation of the netcentric computing system 12.

The web servers 20 are the primary interface to the clients 14 for all interactions with the applications or services that are provided by the netcentric computing system 12. The main task of the web servers 20 is to authenticate the clients 14, establish a secure connection from the clients 14 to the web servers 20 using encrypted messages, and allow applications the clients 14 are using to transparently access the resources of the netcentric computing system 12. The web servers 28 are responsible for accepting incoming HTTP (Hypertext Transfer Protocol) messages and fulfilling the requests. For dynamic HTML (Hypertext Markup Language) page generation, requests are forwarded to the application servers 18.

During operation, static pages, such as help pages, are preferably generated entirely by the web servers 20.

The primary function of the application servers 18 is to provide a link through which the web servers 20 can interact with the clients 14, trigger business transactions, and send back resulting data to the clients 14. A fundamental role of the application servers 18 is to manage the logical flow of transactions and keep track of the state of sessions. The application servers 18 are also responsible for managing all sessions within the netcentric computing system 12. A session is a period of time in which a client 14 is interacting with, and using, a resource of the netcentric computing system 12.

The main purpose of the database servers 16 is to handle an application log. All requests sent to the web servers 20 and application servers 18, as well as their respective responses, are logged in the application log. The application log is preferentially used for traceability. In the preferred embodiment, requests are logged in the application log directly by the application server 18. Those skilled in the art would recognize that any number of data items can be monitored and kept track of in the application log.

As further illustrated in Fig. 2, a second business enterprise location 30 may be connected with the first business enterprise location 28 using an intranet connection 32. Those skilled in the art would recognize that various intranet connections 32 exist and may be used in the present invention. The intranet connection 32 allows the computing resources of the second business enterprise location 30 to be shared or connected with the computing resources available at the first business enterprise location 28. The term intranet connection 32, as used herein, should be broadly construed to include communication devices and software applications as well as various other connection devices used to physically interconnect two or more business networks. Although not illustrated, several other enterprise locations, each containing its own computing resources, may be connected with the netcentric computing system 12 using other intranet connections 32.

In the preferred embodiment illustrated in Fig. 2, the firewall 24 of the first business enterprise location 28 is connected with an Internet connection 32 to a plurality of remote clients 34. The remote clients 34 that are connected to the Internet connection 32 preferentially access data and communicate with the services of the netcentric computing system 12 through the Internet connection 32 using web browser applications that are located and running on the clients 34. The Internet connection 32 gives the remote clients 34 the ability to gain access to applications, information and data content that may be located on the

database server 16, the application server 18 and the web server 20, preferably by means of the web server 20.

As used herein, the term Internet connection 32 should be broadly construed to include any software application and hardware device that is used to connect the remote clients 34 and the servers 16, 18, 20 with an Internet service provider (not illustrated) that establishes a connection to the Internet. Those skilled in the art would recognize that the remote clients 34 and the servers 16, 18, 20 may establish an Internet connection 32 using one of several methods known in the art. For the purpose of the present invention, it is only important to understand that the remote clients 34 and servers 16, 18, 20 are respectively connected with each other through the Internet connection 32.

For a detailed discussion of the technical architecture of the netcentric computing system 12 as well as netcentric computing systems 12 in general, refer to co-pending U.S. patent application Serial Number _____ entitled Architectures For Netcentric Computing Systems, which was filed on _____ and is hereby incorporated by reference in its entirety.

Referring back to Fig. 1, the development architecture 10 includes a common user interface 40, a process management tool 42, a personal productivity tool 44, a quality management tool 46, a systems building tool 48, an environment management tool 50, a program and project management tool 52, a team productivity tool 54 and an information management tool 56. The purpose of the development architecture 10 is to support the tasks involved in the analysis, design, construction and maintenance of business systems used on the netcentric computing system 12, as well as the associated management processes. The development architecture 10 provides tools and resources that are required for developers to operate, design and maintain the netcentric computing system 12.

The common user interface 40 of the development architecture 10 provides a common launching place for all of the tools in the development architecture 10 to make it integrated and consistent. All of the tools that are used in the development architecture 10 are presented to the developer on the clients 14, 34 via a single view of the entire environment. The common user interface 40 is used by the servers 16, 18, 20, preferentially web server 20, to provide a graphical user interface (GUI) to the clients 14, 34 that allows users to access the tools of the development architecture 10 easily and consistently. Applications that support the common user interface 40 are known as window managers (e.g., Microsoft Windows, Presentation Manager and Motif).

The common user interface 40 provides several capabilities to the netcentric computing system 12. As set forth above, the common user interface 40 provides an interactive and graphical interface to users on clients 14, 34. Applications that run on the netcentric computing system 12 are predominantly graphical in nature, typically making use of the Windows operating system. As such, graphical tools are essential, in that developers of netcentric computing systems 12 are constantly multi-tasking and therefore windowing systems are the only practical way for a developer to manage and coordinate work activities on the clients 14, 34.

The preferred common user interface 40 is also capable of shielding system-level details from developers on the clients 14, 34. This allows developers to pick the service desired by selecting it from a menu or an equivalent action on the clients 14, 34. As such, the common user interface 40 that is used in the development architecture 10 is also capable of automating simple details such as naming files and manipulating directories on the servers 16, 18, 20.

Another aspect that the preferred common user interface 40 supports in the development architecture 10 is the ability of developers to handle multiple, concurrent activities on clients 14, 34. Developers should not be restricted to a single development task on clients 14, 34. The common user interface 40 allows developers to simultaneously execute multiple applications and facilities from a respective client 14, 34 without having to close out or end another application on the client 14, 34.

The common user interface 40 also provides access to files, code, and tools, for example, based on privileges defined for each user of the netcentric computing system 20. As such, the preferred common user interface includes security services to keep unwanted or unauthorized users from accessing files, code and tools used in the development architecture 10. The common user interface 40 also provides interactive, context-sensitive help to the users on clients 14, 34. Architecture teams should be encouraged to deliver their API documentation, standards documentation and procedures in an online help format. Today's developers are very accustomed to searching and browsing for answers at the point of need; paper documentation is rarely used.

The process management tools 42 are used to integrate the development architecture 10 by providing tool-to-tool communication and work flow management. Tool-to-tool communication integrates tools by enabling information, in the form of short messages, to be passed from one tool to another. As such, the process management tools 42 provide structure and control over the development process as a whole (e.g., the methodology, procedures, and

supporting processes). As such, they provide for deeper integration than just graphical or desktop integration. The process management tools 42 integrate existing and future tools, both package and custom; provide inter-tool communications where appropriate; and integrates tools with the development repository (discussed later).

The process management tools 42 implementation may vary greatly in complexity. At the simple end is a menu system that presents a single user with the recommended development tasks and can launch the tool appropriate for the selected task. At the high end is a custom, complete work flow implementation that integrates the efforts of different development groups and enforces the project methodology (e.g., it supports the correct sequencing of tasks including reviews and signoffs).

As illustrated in Fig. 1, the personal productivity tools 44 of the development architecture 10 are applications or groups of applications that are used for miscellaneous single-user activities commonly encountered on a project (e.g., writing memos, preparing presentations, and performing simple what-if analyses). Referring to Fig. 3, these tools are typically applications oriented toward individuals rather than teams (i.e., no collaboration is required) and typically consist of spreadsheet applications 60, graphic applications 62, word processor applications 64 and personal calendar applications 66.

As set forth above, the personal productivity tools 44 preferentially include a spreadsheet application 60. Developers should have the ability to access and create spreadsheet data which is used in each phase of the development process. Spreadsheet analysis may be used to analyze reports from a development repository (discussed later) to view test data/results, to perform what-if analyses for impact analysis or estimating, and to assist in modeling such system aspects as performance. Those skilled in the art would recognize that various spreadsheet applications 60 exist and may be used in the preferred development architecture 10. Examples of spreadsheet applications 60 that may be used include Lotus 1-2-3 and Microsoft Excel.

The personal productivity tools 44 also preferentially include graphics applications 62. The graphics applications 62 are most commonly used to prepare presentations and supporting graphics for documentation. The graphics applications 62 may be standalone tools, but are much more useful when they can be integrated directly with the repository or at least the spreadsheet package to allow graphical presentation of information (such as productivity information and quality statistics). Those skilled in the art would recognize that various graphics applications 62 exist and may be used in the preferred development

architecture 10. Examples of graphics applications that may be used include Microsoft PowerPoint, Lotus Freelance and CorelDraw.

Preferably, the personal productivity tools 44 also includes a word processor application 64. Word processor applications 64 provide basic forms and utilities that can be used (e.g., a form letter or memo template) by developers to document project information. Those skilled in the art would recognize that various word processing applications 64 exist and may be used in the presently disclosed development architecture 10. Examples of word processing applications 64 that may be used include AmiPro, Microsoft Word, and WordPerfect.

Those skilled in the art would also recognize the various personal calendar applications 66 exist and may be specially designed for use in the preferred netcentric computing system 12. Personal calendar applications 66 allow users to manage contacts and record various types of calendar information. The personal calendar applications 66 are capable of generating reminders in the form of instant messages, E-mail messages, text pages, etc. Those skilled in the art would recognize that various features may be included in the preferred personal calendar application 66 and are herein envisioned.

Referring to Fig. 1, the preferred development architecture 10 also includes quality management tools 46. Quality management tools 46 are used to ensure that an agreed-on level of quality in the netcentric computing system 12 is reached. These tools also provide information and process for improving quality in the netcentric computing system 12 over time. Quality management tools 46 provide the plan, the measurement, and the feedback for improvement to meet the quality objectives of a project. Referring to Fig. 4, the preferred quality management tools 46 may be selected from the group consisting of quality function development tools 68, measurement and metrics tools 70, statistical process control tools 72 and continuous improvement tools 74.

The quality function deployment tools 68 are developed around the quality plan for the project or the organization. As such, those skilled in the art would recognize that the exact functions and nature of these applications will vary from organization to organization. The preferred quality function deployment tools 68 preferentially focus on the quality objectives that are important for a project. These are expressed in measurable terms whenever possible. For example, the quality function deployment tools 68 can monitor for reliability (in defects per function point), usability (user training or overall productivity),

efficiency (use of systems resources), and maintainability (cost/time to correct problems and provide added functions).

The quality function deployment tools 68 can also be used to define input and output (I/O) criteria for each development phase. This is typically integrated with the development methodology and defines sufficiency criteria for moving from one phase of a project to the next. These criteria are important to ensure that all necessary documentation for a phase has been created and is of the expected quality before starting another phase. This helps reduce rework due to miscommunications or misunderstandings.

The preferred quality function deployment tools 68 are also used to perform identification and definition of the types of test, verification, and validation activities that are to be carried out during testing of different aspects of the netcentric computing system 10. This includes a description of the activities, what they apply to (e.g., validate a functional specification), and when they should occur (e.g., before beginning technical design). The preferred quality function deployment tools 68 are also designed to assign specific responsibilities for quality activities. For instance, the application can be used to determine who is responsible for reviews and tests of the various development components, who has responsibility for configuration management and change control, and who has responsibility for defect control and corrective action. For smaller projects, this responsibility may be spread across the individual developers or teams; on larger projects, responsibility may be assigned to a specific quality team that interacts with the individual development teams.

The quality function deployment tools 68 are also used to reveal, document, and prioritize the requirements for systems and applications under development that are to be deployed on the netcentric computing system 12. Based on these requirements, it is possible to define meaningful goals for product quality along different dimensions (e.g., maintainability, complexity, and performance).

The measurement and metrics tools 70 are an important part of the quality management tools 46 because they provide operational definitions of quality attributes. These applications or functions provide an operational definition for a method of sampling, testing, and determining whether a work product meets a given criterion. With the operational definitions, different users can use the measurement and metrics tools 70 to agree that a product objectively meets a requirement, or that a process has been improved by a measurable amount. To fine-tune the development process, it is necessary to be able to measure the important quality attributes. These measurements will evolve as software

engineering matures and netcentric computing systems 12 expand, but sample items that the preferred measurement and metrics tools 70 monitor include: the average number of defects per design packet at the moment construction starts; the average number of defects per program at the time of its first migration to product test; system availability and causes of downtime in the netcentric computing system 12; time needed for a new user to learn to use a function of the netcentric computing system 12; user error rate per function; and maintainability in terms of time to fix a defect or to add new functions.

For the measurement and metrics tools 70 to capture this information, the tools or applications used to perform a function must provide support for capture of quality statistics. For example, the source code management toolset can allow information to be specified about reasons for a change, and the stage the component had reached (e.g., initial construction, product test, and production). This information could be stored and placed in a quality statistics part of the development repository for later reporting.

The statistical process control tools 72 monitor and relate to the methodology, work flow, and tools usage in the netcentric computing system 12. As such, the statistical process control tools 72 ensure that quality gets built into the end product from the beginning of development. The applications that are designed for the statistical process control tools 72 preferentially implement standards and procedures pertaining to quality assurance of the process, describe how to use simple tools, such as templates and checklists, and document the mandatory outputs from each work process. Other procedures applications can perform include common tasks such as design reviews and code reviews.

The continuous improvement tools 74 include applications that capture feedback on the quality process and can take actions to improve it or notify individuals, by e-mail for example, if necessary. The continuous improvement tools 74 also preferentially include applications that can create an electronic suggestion mailbox to receive suggestions from various users, employees, as well as public users of the netcentric computing system 12. Those skilled in the art of programming would recognize that various applications may be used in the continuous improvement tools of the quality management tools 46.

As illustrated in Fig. 1, the preferred development architecture 10 also includes system building tools 48. The system building tools 48 comprise the core of the development architecture 10 and are used to design, build and test the overall functionality of the netcentric computing system 10. As such, the systems building tools 48 are the most important part of the development architecture 10. The system building tools 48 include

applications that are used by the development team to capture the system requirements, the functional design, the detailed design decisions, the detailed coding and testing and to manage the resulting (frequently large number) components of the netcentric computing system 12.

Referring to Fig. 5, the preferred system building tools 48 may be selected from the group consisting of analysis and design tools 76; reverse engineering tools 78; construction tools 80; testing tools 82; and configuration management tools 84. The system building tools 48 are the core of the development architecture 10 and are used to design, build, maintain and monitor applications used on the netcentric computing system 12. The analysis and design tools 76 are used to capture the requirements for the application being developed, to analyze and prioritize them, and to transform them into a functional definition and then into a detailed technical definition suitable for construction. In other words, the analysis and design tools 76 help specify "what" a system needs to do, and design tools help specify "how" a system will implement the "what." A number of the analysis and design tools 76 are typically part of an I-CASE (integrated computer-aided software engineering) package such as FOUNDATION, Paradigm Plus, and Software through Pictures.

In the preferred embodiment of the present invention, the analysis and design tools 76 may be selected from the group consisting of data modeling tools, process modeling tools, event modeling tools, database design tools, application logic design tools, presentation design and modeling tools, communication design tools, performance modeling tools and object and component modeling tools. The data modeling tools provide the capability to graphically depict the logical data requirements for the system on the clients 14, 34. Typically, a tool for data modeling supports diagramming entities, relationships, and attributes of the business being modeled on an entity-relationship diagram (ERD). The key difference from a traditional data modeling tool is the ability to capture information necessary for making data distribution decisions (e.g., ownership of data and frequency of access /manipulation by location).

The process modeling tools provide the capability to depict (preferably graphically on the clients 14, 34) the business functions and processes being supported by a system of the netcentric computing system 12, including, for example, tools that support documenting process decomposition, data flow, and process dependency information. As with the data modeling tools, the main difference in these tools for netcentric is the ability to capture the information necessary to make process placement decisions. For example, where the process

needs to occur (on a mobile personal computer, at a stationary workstation), the type and volume of data it requires to perform the function, and the type of function (user interaction, reporting, batch processing).

The event modeling tools provide the capability to depict the business events and associated responses of the netcentric computing system 10 on the clients 14, 34. A variety of tools and techniques can be used for event modeling, including word processors, to develop simple textual lists of events and diagramming tools to show events and responses on respective clients 14, 34. The event modeling tools inherently are closely tied to the process modeling tools because events result in a response generally described by a process.

The database design tools provide users on clients 14, 34 with the capability to capture the database design for the netcentric computing system 12. The tools enable the developer to illustrate, for example, the tables and file structures that will be physically implemented from the logical data requirements. The tools also capture the definition of data elements, indexing decisions, foreign keys and referential integrity rules. As with the data modeling tools, the key difference in the database design tools for client/server and netcentric computing systems 12 is the ability to capture data distribution decisions and to provide support for creating schemas for multiple database products. For example, the standalone laptop database management system (DBMS) may be different from the work group and enterprise DBMS choices (such as SQL-Anywhere on the laptop and DB2 at the enterprise).

The application logic design tools provide users on clients 14, 34 with the capability to depict the logic of the application, including application structure, module descriptions, and distribution of function across various nodes of the netcentric computing system 12. A variety of tools and techniques can be used for application logic design, including structure charts, procedure diagrams (module action diagrams), and graphics packages to illustrate distribution of function across client and server.

The presentation design and prototyping tools provide users on clients 14, 34 with the capability to depict the presentation layer of a particular application, including screens, windows, reports, and dialog flow. Tools in this category include window painters, report painters and dialog flow diagrammers.

Window painters let the developer use clients 14, 34 to design windows for applications using common GUI window controls and application variables. The behavior associated with a window can either be captured directly in the window painter tool or using a deliverable known as a CAR (control, action, response) diagram. Frequently specified as a

matrix or structure chart, the CAR diagram captures the response to an action taken on a particular control--for example, what to do when the user exits a field or clicks a button.

Report painters let the developer use clients 14, 34 to design the report layout interactively, placing literals and application data on the layout without specifying implementation details such as page breaks. Typical window, screen, and report painters also generate the associated application code or a structure in which remaining code can be placed during construction. In addition, many window painters provide the capability to rapidly prototype user interfaces.

Prototyping tools allow developers to follow a more iterative functional design approach, which is important when dealing with developers and users that may be new to the GUIs typical of client/server and netcentric systems. In addition, given the responsive nature of a GUI, prototyping becomes an effective way of clearly communicating how the system appears to the user, by allowing developers to view and interact with applications from the clients 14, 34 before final implementation. Another aspect the prototyping tools provide is enabling developers to rapidly build and modify screens and windows. Beyond this basic requirement, some prototyping tools may support the specification and prototyping of the dialog flow (e.g., linking windows), simple data interaction such as validation, or more complex data interaction such as the ability to insert, save, and transfer data between screens.

Examples of window painting and prototyping tools include Sybase's PowerBuilder, Microsoft's Visual Basic, SQLWindows from Centura Software Corp., and Visual Edge's UIM/X. Examples of report painters include Sybase's SQR and Crystal Reports by Seagate.

The communication design tools allow designers to specify the contents of an exchange and define the "contract" of the exchange in terms of the processing to be performed, the expected preconditions, and the handling of error or unexpected conditions. The communication design tools can also provide a generation capability for the code or common structures required in construction to send and receive the message. After the fundamental communication paradigms have been chosen (message passing, remote procedure call, structured query language-based), each exchange must be specified in detail to take into account the detailed design of the sending and receiving modules (clients 14, 34, services, subroutines, functions) and to lay the basis for more refined performance modeling. Multiple tier netcentric computing systems 12 can only be built efficiently if the interfaces between the tiers are precisely specified.

The performance modeling tools support the analysis of the system performance of the netcentric computing system 12. An application that generates a simple spreadsheet may

be suitable in some well-known and understood environments, but dedicated performance or simulation modeling applications are preferentially used for any applications with high transaction volumes or complex multi-tier architectures involving several platforms.

In netcentric computing systems 12, the performance of the network is critical. However, it is impossible to guarantee the performance of an application once it has passed by the ISP (Internet Service Provider) over the Internet connection 32 to the clients 14, 34. Therefore, the preferred performance modeling tool is also able to model the performance to the ISP, as well as provide the ability to do "what-if" scenarios for the network design and security implications.

The object and component modeling tools provide specific applications for creating object and component models that can be used to automate the component design process, as well as create and document the component model. Some of these tools are also capable of generating code.

As previously set forth, the system building tools 48 preferentially include a reverse engineering tool 78. The preferred reverse engineering tool 48 may be selected from the group consisting of a system structure analysis tool, an extraction tool, a repository population tool and a restructuring tool. As known in the art, reverse engineering is a set of techniques used to assist in reusing existing system components-either directly (e.g., code/modules) or indirectly (e.g., design rules or algorithms, and record layouts) on the creation of new applications. The reverse engineering tools 78 are used to streamline the development process. Although the reverse engineering tools 78 cannot completely automate the analysis process, they can reduce the amount of manual effort needed, and significantly lessen the amount of non-value-added automatic activities such as "find all the places in a program that affect the value of a given variable."

The reverse engineering tools 78 preferentially include a system structure analysis tool that is used by a developer to identify requirements for a new system from the capability and design of a legacy system. These applications enable the developer to interactively and graphically navigate through the legacy system, analyzing such system characteristics as system structure, module flow, flow of control within a module, calling patterns, complexity, and data and variable usage.

The system structure analysis tools can also provide cross-reference listings or graphical representations of control or data flows to users on clients 14, 34. These tools are most effective when they are used to find and understand the business rules implemented by a

system (that may no longer be documented) to provide comparable features in a new system. Examples include VIA Insight, VIA Renaissance, and Compuware PATHVU.

The preferred reverse engineering tools 78 also include an extraction tool. The extraction tool, in conjunction with a repository population tool, provides the developer with the capability to reuse selected portions of a legacy system. The preferred extraction tool employed in the netcentric computing system 12 will read and extract information from source code, screens, reports, and the database. The most common information the extraction tools extract from a legacy system is data: record or table structure, indexes, and data element definitions. Although it is difficult to extract functions and processes from source code, source code containing complex algorithms is another candidate for extraction. Example extraction tools that could be used include Adpac's PMSS and Viasoft's Alliance.

The preferred reverse engineering tools also include a repository population tool. The repository population tools load the information from the extraction tools and the structure analysis tools into a development repository, which is preferentially located on a respective server 16, 18, 20. These tools convert the information from the legacy system into the syntax of the development repository of the system building tools 48.

The preferred reverse engineering tools 78 also include at least one restructuring tool. The restructuring tools are not analysis tools like the previous categories of reverse engineering tools, but rather design and construction tools. The restructuring tools enable the developer to rebuild a legacy system rather than replace it. Examples of this type of process include restructuring spaghetti code into structured code, replacing GOTO's with a PERFORM construct, streamlining the module calling structure, and identifying and eliminating dead code. These are most often encountered on projects that are rehosting one or more applications to a netcentric computing system 12 and want to upgrade the existing code to make it more maintainable and extend its useful life. Examples of restructuring tools for COBOL programs is Compuware's PATHVU, Compuware's RETROFIT, Knowledgeware's Inspector, Knowledgeware's Recoder, and IBM's SF.

Referring to Fig. 5, the construction tools 80 of the system building tools 48 are used to program, or build, applications: clients 14, 34 and server 16, 18, 20 source code, windows or screens, reports, and databases. Sophisticated tools support the rapid creation of client/server systems are readily available. These visual programming tools (e.g. PowerBuilder and Visual Basic) simplify the creation of 2-tier client/server systems by providing tools and languages geared towards user interface development while also providing graphical controls that link directly to relational data sources.

Building on these successful 2-tier client/server tools, several vendors have advanced their products to support multi-tier development. Tools such as Microsoft's Visual C++ provide sophisticated end to end development environments complete with debugging across multiple tiers. However, these tools are often aimed at a specific vendor's platform or technology. The netcentric developer's construction toolset typically contains a set of point tools that support the creation of user interfaces using markup languages such as HTML, DHTML and XML that contain embedded scripts for controlling Java applets that interact over the network with business logic constructed from reusable components.

There are five categories of construction tools 80 used in the preferred system building tools 48. Except for some specific utilities, it is unlikely that these tools will be purchased separately. All of the major tool vendors package these tools into an Integrated Development Environment (IDE), many of which provide links to the configuration management tools 84. The construction tools 80 are preferentially selected from the group consisting of publishing and page mark-up tools, source code editor tools, generation tools, compiler/linker/interpreter/debugger tools and construction utility tools.

The publishing and page markup tools allow developers to create individual web pages that are displayed in a web browser on a respective client 14, 34, along with the links that allow the end user to navigate between web pages. Publishing and page mark-up tools such as Microsoft's Front Page are able to manage large complex webs of HTML documents, providing visual analysis of broken links and page hierarchies. Netcentric computing systems 12 use web browsers for the user interface and thus requires tools that support the creation of documents in HTML, DHTML and XML. These tools also support the creation of scripts (e.g. VBScript, JavaScript) within the HTML. Script languages allow the developer to code behavior into the page, to support basic validation and capture of data.

The source code editor tools are used to enter and edit source code for a particular application. Typically, editors are provided by an IDE, but many IDEs allow editors to be replaced by popular and more powerful editors such as Brief. Most editors provide source highlighting and integration with online help systems. Within the IDE, the editor is coupled to the compiler to provide incremental syntax checking, rapid compilation, and the ability to run and test the application without having to leave the editing environment (e.g., C++ development environments from Borland, Microsoft, and IBM).

The generation tools are automated tools that generate some component of the application: source code, common structures, windows, reports, and the database definition. These applications convert the application design into some form of source code. Some

common types of generation tools include procedural code generator tools, shell generation tools, and data design language and data manipulation language generator tools.

The procedural code generator, also known as source code generators, take a pseudo-code specification of a module and generate a module in the appropriate programming language. Alternatively, the procedural code may be specified in the development repository using the target programming language (this eliminates an additional language that would have to be learned by a developer). This approach is common in I-CASE products that allow logic to be generated for multiple platforms (such as FOUNDATION and IEF).

Shell generation tools are used when it is not feasible or desirable to specify detailed code within the development repository. As such, a shell of a module can be generated with the shell generation tools with markers for where module specific code should be entered by a programmer. These markers are frequently encountered in window painting tools that can generate the modules required to implement the window with all the housekeeping code already in place. Visual C++ from Microsoft is an example of a tool that offers such a capability--it generates the shell code for windows painted in the environment and allows the programmer to add the business logic at specified drop points.

Data design language (DDL) and data manipulation language (DML) generator. Based on the data and access definitions specified in the repository, these would generate the schema definition for the appropriate DBMS, and the structured query language (SQL) and support code for performing the database I/O. DDL generators are frequently included in some of the I-CASE offerings discussed earlier. DML generators are either custom-developed for a project or may be built on top of general-purpose query tools (such as Q&E or report writers). In the latter case, the query tool is used to build the query and the resulting SQL is copied into the appropriate module.

Compiler/linker/interpreter/debugger tools are usually part of an IDE - it is rare today to be able to purchase a standalone compiler (the exceptions are midrange and mainframe platforms, although products such as IBM's Visual Age are also becoming popular on these platforms).

A compiler/linker converts source code to executable code and packages it into a runtime module. Third-generation languages such as C, C++ and COBOL are all compiled languages. An interpreter executes the source code directly (or indirectly through

a pseudo-compiled intermediate representation). Java and Visual Basic are the best known interpreted languages, although the latest versions can also be compiled.

A source code debugger (sometimes known as a symbolic debugger) is used to step through a program or module at the source code level (as opposed to the machine code level). Although commonly used for debugging programs, source code debuggers are also effective in supporting component testing, because variables can be changed to cause logic to be executed that might otherwise be difficult to simulate using external data (e.g., time sensitive logic, logic for handling I/O hardware failures). Debuggers are typically included in an IDE.

As previously set forth, the preferred construction tools 80 also include construction utilities. Construction utilities are an assortment of tools that facilitate the construction process. The construction tools 80 provided as part of an IDE are normally adequate for use by individual developers. Developers responsible for the overall build of a large system will require standalone tools, particularly for managing and building all of the source code. The construction utilities provide applications that help manage and build all of the source code associated with a particular project. Preferentially, the construction utilities include a MAKE utility, a portability checker, an application shell and a profiler.

The MAKE utility uses information (contained in a makefile) about the dependencies between modules of the system to automate compilation and linking of an application. By examining timestamps and dependencies the MAKE utility streamlines the build process because only those components dependent on the change are recompiled/linked. The makefile can either be hand-coded or it can be generated based on information in a repository. Generation of the makefile tends to eliminate troublesome mistakes, assuming that the development repository administrator has done a good job of maintaining the integrity of the development repository. MAKE utilities are available either as standalone applications (such as NMAKE from Microsoft or MAKE from Opus) or built into an integrated workbench environment (e.g., the "project" concept in Microsoft Visual C++ or Visual Basic).

The portability checker is a utility that checks compliance with basic portability standards--particularly with programming standards that ensure portability across platforms (ANSI C, POSIX). These utilities are typically used only on the C and C++ programming languages because these are the dominant languages available on numerous platforms and operating systems. Portability checking is sometimes built in to the compiler and can be

turned on with a switch or flag specified at compile time or can be performed by separate tools (e.g., the lint utility available under UNIX).

The application shells depict basic application functions for common module types that can be used as a starting point (a shell) for design and programming. The application shells can be used in detailed design and programming phases of the development life cycle to enable designers and programmers to focus more on the essential business logic and spend less time worrying about structural aspects that can be solved once for everyone (e.g., a structure for dealing with the paging of large lists and an approach for handling data passing between modeless windows). Furthermore, application shells are a good mechanism for enforcing standards as they typically embody them and thus the programmers get the standards for free. Because application shells are usually project specific, their initial creation is done manually (rather than being generated).

A profiler provides information and metrics needed to monitor and improve code quality and maintainability and make suggestions on how to package the code into modules and libraries for performance optimization. Several different types of profilers may be developed and used in the presently disclosed development architecture 10.

The system building tools 48 also preferentially include a testing tool 82. Testing is the process of validating that the gathering and transformation of information has been completed correctly and to the expected quality level. Testing is usually considered the process that makes sure there are no bugs in the code. But in a broader sense, testing is about making sure that the netcentric computing system 12 does what it is expected to do (i.e., meets the requirements specifications) at an acceptable quality level (e.g., acceptable numbers of defects per function point, or defects per module). Those skilled in the art would recognize that various testing tools 82 may be designed and used in the present invention, depending on the needs and requirements of each netcentric computing system 12.

The preferred testing tools 82 include stubs and drivers that are used to test various components of an application or architecture before a complete set of components is available. These are generally custom-coded as part of the component testing effort. Stubs emulate subroutines or external functions in a minimal fashion--that is, they basically return with some sample data and the various return code values (e.g., successful and failed). They are useful for testing a module when the modules it calls are not yet ready or available for testing. Harnesses and drivers call up a module and emulate the context in which the module will be called in the production environment.

As previously set forth, the preferred system building tools 48 also include configuration management tools 84. The configuration management tools 84 handle the management of components in the netcentric computing system 12 to ensure that the components collectively satisfy the given requirements of the netcentric computing system 12. "Configuration" designates a set of components in a given environment satisfying certain requirements. The configuration management tools 84 ensure that consistency is maintained over time, even with changes to the components. The components of the netcentric computing system 12 are typically hardware, system software, and application components (such as source code, executable modules, load libraries, database DDL, and scripts or job control language), together with their documentation. The development architecture 10 also includes test data, test scripts, and other components that must be aligned with a given version of the configuration.

Version control and compatibility of components are key considerations when managing components of a netcentric computing system 12. Version control applies to all types of components, not just application components. In case incompatibilities are discovered, it must always be possible to "roll back" to a previous consistent state--that is, to revert to an earlier version of one of more components. To do this, it is necessary to know which versions are compatible. It must be possible to define releases of a configuration--a list of version numbers, one for each component, which together form a consistent configuration. The configuration management tools 84 provide this functionality to ensure proper versions of applications are being executed on the netcentric computing system 12.

In the preferred embodiment, the configuration management tools 84 for the development architecture 10 preferentially includes version control tools and migration control tools. Version control tools control access to source code and other development components as they are developed and tested in the netcentric computing system 12. They typically allow releases to be defined and multiple "snapshots" (i.e., the versions of all the components in the release) to be taken and maintained to facilitate rolling back to earlier releases if necessary. Examples of version control tools include Intersolv's PVCS and the UNIX Source Code Control System (SCCS).

Migration control tools control multiple versions of source code, data, and other items as they are moved across different environments of the netcentric computing system 12. The source code migration control tools manage multiple versions of source code to ensure that changes are applied in the proper environment and that thoroughly tested modules are subsequently migrated to the next environment. Data migration control tools manage

As previously set forth, the preferred system building tools 48 also include configuration management tools 84. The configuration management tools 84 handle the management of components in the netcentric computing system 12 to ensure that the components collectively satisfy the given requirements of the netcentric computing system 12. "Configuration" designates a set of components in a given environment satisfying certain requirements. The configuration management tools 84 ensure that consistency is maintained over time, even with changes to the components. The components of the netcentric computing system 12 are typically hardware, system software, and application components (such as source code, executable modules, load libraries, database DDL, and scripts or job control language), together with their documentation. The development architecture 10 also includes test data, test scripts, and other components that must be aligned with a given version of the configuration.

Version control and compatibility of components are key considerations when managing components of a netcentric computing system 12. Version control applies to all types of components, not just application components. In case incompatibilities are discovered, it must always be possible to "roll back" to a previous consistent state--that is, to revert to an earlier version of one of more components. To do this, it is necessary to know which versions are compatible. It must be possible to define releases of a configuration--a list of version numbers, one for each component, which together form a consistent configuration. The configuration management tools 84 provide this functionality to ensure proper versions of applications are being executed on the netcentric computing system 12.

In the preferred embodiment, the configuration management tools 84 for the development architecture 10 preferentially includes version control tools and migration control tools. Version control tools control access to source code and other development components as they are developed and tested in the netcentric computing system 12. They typically allow releases to be defined and multiple "snapshots" (i.e., the versions of all the components in the release) to be taken and maintained to facilitate rolling back to earlier releases if necessary. Examples of version control tools include Intersolv's PVCS and the UNIX Source Code Control System (SCCS).

Migration control tools control multiple versions of source code, data, and other items as they are moved across different environments of the netcentric computing system 12. The source code migration control tools manage multiple versions of source code to ensure that changes are applied in the proper environment and that thoroughly tested modules are subsequently migrated to the next environment. Data migration control tools manage

multiple versions of the database and its data to ensure that accurate data and structure are maintained in the environment and that versions of application code and database are deployed consistently. Types of data that would be migrated include base codes data or other reference data (e.g., a state code table or valid order code table) and converted business data. Other migration control tools manage other types of system objects to ensure that a complete version of all components reside in the production environment (e.g., architecture support files, test definitions, and scripts).

Referring back to Fig. 1, the preferred development architecture 10 includes an environment management tool 50. The environment management tools 50 monitor performance, provide help desk support, manage and distribute changes to the development architecture 10, administer the environment and track and plan capacity. Adopting a structured approach to environment management, applying the same principles to development as to production, has several advantages. It provides high-quality support for developers. In addition, environment management can provide significant experience with the operations management tools in an environment that is generally smaller and carries lower risk than the full production environment. Environment management facilitates the tuning of the production support approach before production roll-out. The approach is refined from experiences using it to support the development team.

Referring to Fig. 6, the preferred environment management tools 50 include service management tools 86, system management tools 88, managing change tools 90 and service planning tools 92. The environment management tools 86 support different functional and technical requirements of development teams, and include tools that support the various stages of the lifecycle of an application used on the netcentric computing system 12. The service management tools 85 define and manage to an agreed-on level of service, including service-level agreements, information gathering to check against the service-level agreements, and help desk support for the developer community. The system management tools 88 manage the development architecture 10. These tools provide support for managing security, starting up and shutting down the development architecture 10, and performing backups of files and applications.

The managing change tools 90 are used for making, tracking, and distributing changes to the development architecture 10. The most common type of change is upgrading of software (system, architecture, or application), but changes to hardware configurations and network configurations must also be supported. The service planning tools 92 support a capacity planning function for the development architecture 10. The environment needs to be

monitored and sufficient lead time allowed to support required capacity changes for shared disk space, server size (e.g., central processing unit size, memory, and number of users), network, and workstations (either the number of workstations or the configuration of the workstations).

Referring to Fig. 1, the program and project management tools 52 provide many key features that assist project planners in planning, scheduling, tracking and reporting on project segments, tasks and milestones. In the preferred embodiment of the present invention, the program and project management tools 52 are differentiated by the ability to support multiple projects, complex functions and adequate performance when supporting multiple concurrent projects. The presently preferred program and project management tools 52 may be selected from the group consisting of planning tools 94, scheduling tools 96, tracking tools 98 and reporting tools 100. Those skilled in the art would recognize that depending on the enterprise's operations, the programming and project management tools 52 may vary from enterprise to enterprise.

The planning tools 94 are tightly linked with the development methodology. The planning tools 94 help in estimating the development effort, defining the project tasks and activities, and identifying the type and quantity of resources required (subject matter experts, architects, designers). When the planning tools 94 have determined estimates and resource requirements, the scheduling tools 96 assist in scheduling the work, identifying dependencies and critical paths, and balancing (level loading) the work across the resources. On an ongoing basis, the scheduling tools 96 also provide administration features that allow tasks to be assigned and reassigned as the project evolves.

The tracking tools 98 provide a mechanism for members of the development team to report time against a particular project plan. This is typically done on a weekly or biweekly basis. The reporting tools 100 provide reporting capabilities to reflect the status of the project against the plan. In the simplest form, the reporting consists of budget and schedule information, such as time spent by member, budget variance, schedule variance, estimates to complete, and planned versus actual results. More advanced tools can provide information on productivity and efficiency. Most project planning and management tools 52 available today provide some capability for each of the above. Examples of these tools include Microsoft Project and ABT Project Manager's Workbench.

Referring to Fig. 1, the team productivity tools 54 are used to make the work cell and project team as a whole more productive within the enterprise. Instead of the software residing on the client 14, 34, the team productivity tools 54 are typically LAN-based and

shared by the project members on clients 14, 34. As such, the team productivity tools 54 are typically located on the servers 16, 18, 20. In the preferred embodiment of the present invention, the team productivity tools 54 are focused on enhancing communication and information sharing within the business enterprise and may be selected from the group consisting of E-mail tools 102, teamware tools 104, publishing tools 106, group calendar tools 108, and methodology browsing tools 110. Those skilled in the art would recognize that several other team productivity tools 54 may be incorporated into the netcentric computing system 12.

An E-mail tool 102 (e.g., Lotus Notes, or Microsoft Exchange) is valuable for sharing such dynamic information as design documents, meeting schedules, project events, and resource availability. Because E-mail tools 102 allow mail to be stored, forwarded, sorted, and filtered dynamically, they improve the quality of communication; they also speed up the flow of information between users. Those skilled in the art would recognize that several E-mail tools 102 may be used in the present invention.

Teamware tools 104 allow groups of people to share information easily. The teamware tools 104 typically provide a forum for people with a common interest to share information and ask questions of one another. Depending on the desired environment, the teamware tools 104 forums may include newsgroups, bulletin boards, or databases. What they have in common is the ability to post questions and comments and to search through the existing discussion to see whether the information required is already present. Like E-mail tools 102, the posting and reading of information takes on the look of a mail letter. Unlike E-mail tools 102, however, the "letters" are openly available to everyone with access to the bulletin board and are saved for an extended period of time.

The publishing tools 106 allow individuals to create and print anything from single deliverables or specs all the way through the complete set of documentation for the system. Because documentation may be spread over several hardware platforms, and because it may reside in different libraries in different formats and may have to be printed using different tools, it is important to ensure that any chosen publishing tools 106 can inter-operate or integrate to allow aspects such as common headers and footers and consecutive page numbering to be handled without overly intensive manual involvement.

The preferred team productivity tools 54 also includes group calendar tools 108 that are used for scheduling purposes and routine calendar tasks. These tools allow users to schedule important items, generate reminders of deadlines, and various other functions, commonly provided in group calendar tools 108. Those skilled in the art would recognize

that various group calendar tools 108 may be used in the preferred development architecture 10 for the netcentric computing system 12.

The methodology browsing tools 110 are used in the team productivity tools 54 to allow users to browse, from clients 14, 34, various types of documents and files located on the netcentric computing system 12 that are associated with the project methodology. For instance, viewing the overall development lifecycle, descriptions of specific tasks and deliverables, task considerations and other tasks that are related or dependent.

Referring to Fig. 9, the information management tools 56 include a development repository 112, a folder management tool 114 and a repository management tool 116. In one preferred embodiment of the present invention, the information and data for applications is stored in the development repository 112 on such devices as hard drives, CD-ROMS, and magnetic tapes. However, those skilled in the art would recognize that various other storage devices may be used as well.

The information management tools 56 share a common repository of development objects, design documents, source code, and test plans and data in the development repository 112. Ideally, the development repository 112 would be a single database with an all-encompassing information model. The development repository 112 is built by integrating the repositories of the different development tools through various interfaces. Specific tool vendors may also build part of the integrated repository by integrating specific products.

The preferred development architecture 10 also includes a development repository 112. The development repository 112 is the communication backbone of the development architecture 10, making it easy to share information between people working on different processes. The development repository 112 stores design, construction, and maintenance information, such as window layouts, processing specifications, code fragments, and references to source code files. By storing this information in the development repository 112, several benefits can be realized in the development architecture 10.

The use of the development repository 112 is made an integral part of designers' and developers' daily activities. The development repository 112 is a tool that assists the team, but even simple development repositories 112, such as a well-managed set of shared directories on a network server 16, 18, 20 can provide significant benefits. The key to success is ensuring that the development repository 112 is at the heart of the development processes, remaining intact and populated with current information.

By providing a common "template" for the content and format of design information, developers can create consistent specifications. In addition, by providing a "pool" of

common definitions (especially for such low-level objects as data elements, table/record definitions, windows, and reports), the development repository 112 facilitates consistent use and interpretation and, in some cases, reuse.

For example, by providing a common place for element definitions, and including such display information as literals and field size, windows and reports are more likely to integrate with the database definition and more likely to display or interact with the end user in a consistent manner (field validation, the literal to the left of the field, the length of the field). Without this information in the development repository 112, it would be up to individual developers to seek out the "common" information and apply it appropriately while they define their windows and reports. Consistent capture and organization of information makes it much easier for more automation (e.g. code generators) to be provided in the future.

The development repository 112 cannot force reuse to occur, but it is a building block on which to start a program of reuse. Because information about low-level (elements) and high-level (functions, subsystems) entities is stored in the development repository 112, it is a logical place to begin looking for reusable building blocks for developers. This reuse commonly happens within a team on a project but can also happen across teams within a project and eventually across projects.

The folder management tools 114 allow users to manage documents and files within the development repository 112 of the information management tools 56. In addition, the repository management tools 116 monitors and manages the contents of the development repository 112. Those skilled in the art would recognize that the exact functions of the folder management tools 114 and the repository management tools 116 will vary, depending on the needs of each particular netcentric computing system 12.

While the invention has been described in its currently best known modes of operation and embodiments, other modes and embodiments of the invention will be apparent to those skilled in the art and are contemplated. For other features, advantages and combinations of the present invention refer to U.S. provisional application Serial No: 60/156,962, which is herein incorporated by reference in its entirety.

What is claimed is:

1. A development architecture for a netcentric computing system, comprising:

at least one server connected with a client;

wherein said server provides a common user interface between said server and said client, said server also providing at least one process management tool, at least one personal productivity tool, at least one quality management tool, at least one system building tool, at least one environment management tool, at least one program and project management tool, at least one personal productivity tool and at least one information management tool for use by said client.

2. The development architecture of claim 1, wherein said common user interface is used by said server to provide a graphical user interface to said client.

3. The development architecture of claim 1, wherein said process management tool allows one of said tools to communicate with at least one other respective tool.

4. The development architecture of claim 1, wherein said personal productivity tool may be selected from the group consisting of a spreadsheet application, a graphic application, a word processing application and a personal calendar application.

5. The development architecture of claim 1, wherein said quality management tool may be selected from the group consisting of a quality function development tool, a measurement and metrics tool, a statistical process control tool and a continuous improvement tool.

6. The development architecture of claim 1, wherein said system building tool may be selected from the group consisting of a analysis and design tool, a reverse engineering tool, a construction tool, a testing tool and a configuration management tool.

7. The development architecture of claim 6, wherein said analysis and design tool may be selected from the group consisting of a data modeling tool, a process modeling tool, a event modeling tool, a database design tool, application logic design tool, a presentation and design tool, a communication design tool, a performance modeling tool and a component modeling tool.

8. The development architecture of claim 6, wherein said reverse engineering tool may be selected from the group consisting of a system structure analysis tool, an extraction tool, a repository population tool and a restructuring tool.

9. The development architecture of claim 6, wherein said configuration management tool include a version control tool and a migration control tool.

10. The development architecture of claim 6, wherein said construction tool may be selected from the group consisting of a publishing and page mark-up tool, a source code editor tool, a generation tool, a compiler/liker/interpreter/debugger tool and a construction utility tool.

11. The development architecture of claim 1, wherein said environment management tool may be selected from the group consisting of a service management tool, a systems management tool, a managing change tool and a service planning tool.

12. The development architecture of claim 1, wherein said program and project management tool may be selected from the group consisting of a planning tool, a scheduling tool, a tracking tool and a reporting tool.

13. The development architecture of claim 1, wherein said team productivity tool may be selected from the group consisting of a E-mail tool, a teamware tool, a publishing tool, a group calendar tool and a methodology browsing tool.

14. The development architecture of claim 1, wherein said information management tools includes a development repository, at least one folder management tool and at least one repository management tool.

15. A development architecture for a netcentric computing system, comprising:

a server connected with a client, wherein said server provides a common user interface between said server and said client;

a personal productivity tool that may be selected from the group consisting of a spreadsheet application, a graphic application, a word processor application and a personal calendar application for use by said client and said server;

a quality management tool for assuring that a predetermined agreed upon level of quality is maintained by said netcentric computing system;

a system building tool for designing, building and testing applications on said netcentric computing system;

a environment management tool for monitoring the performance of said netcentric computing system;

a program and project management tool for planning, scheduling, tracking and reporting on project segments in said netcentric computing system;

a team productivity tool for allowing users on said clients to communicate with other users in the netcentric computing system;

a information management tool including a development repository, a folder management tool and a repository management tool;

a process management tool for allowing a respective said tool to communicate with another respective one of said tools in said netcentric computing system.

16. The development architecture of claim 15, wherein said system building tool may be selected from the group consisting of a analysis and design tool, a reverse engineering tool, a construction tool and a configuration management tool.

17. The development architecture of claim 16, wherein said analysis and design tool may be selected from the group consisting of a data modeling tool, a process modeling tool, a event modeling tool, a database design tool, application logic design tool, a presentation and design tool, a communication design tool, a performance modeling tool and a component modeling tool.

18. The development architecture of claim 16, wherein said reverse engineering tool may be selected from the group consisting of a system structure analysis tool, an extraction tool, a repository population tool and a restructuring tool.

19. The development architecture of claim 16, wherein said construction tool may be selected from the group consisting of a publishing and page mark-up tool, a source code

editor tool, a generation tool, a compiler/liker/interpreter/debugger tool and a construction utility tool.

20. The development architecture of claim 16, wherein said configuration management tool include a version control tool and a migration control tool.

21. The development architecture of claim 15, wherein said environment management tools may be selected from the group consisting of a service management tool, a system management tool, a managing change tool and a service planning tool.

22. The development architecture of claim 15, wherein said program and project management tool may be selected from the group consisting of a planning tool, a scheduling tool, a tracking tool and a reporting tool.

23. The development architecture of claim 15, wherein said team productivity tool may be selected from the group consisting of a E-mail tool, a teamware tool, a publishing tool, a group calendar tool and a methodology browsing tool.

24. A method of providing a development architecture for a netcentric computing system, comprising the steps of:

providing a server connected with a client, wherein said server provides a common user interface between said server and said client;

providing a personal productivity tool that may be selected from the group consisting of a spreadsheet application, a graphic application, a word processor application and a personal calendar application for use by said client and said server;

assuring that a predetermined agreed upon level of quality is maintained by said netcentric computing system with a quality management tool;

designing, building and testing applications on said netcentric computing system with a system building tool;

monitoring the status of a project on said netcentric computing system with a environment management tool;

planning, scheduling, tracking and reporting on project segments in said netcentric computing system with a program and project management tool;

allowing users on said clients to communicate with other users in the netcentric computing system with a team productivity tool;

providing an information management tool including a development repository, a folder management tool and a repository management tool;

communicating with one said tool with another respective tool in said netcentric computing system with a process management tool.

25. The method of claim 24, wherein said system building tool may be selected from the group consisting of a analysis and design tool, a reverse engineering tool, a construction tool and a configuration management tool.

26. The method of claim 25, wherein said analysis and design tool may be selected from the group consisting of a data modeling tool, a process modeling tool, a event modeling tool, a database design tool, application logic design tool, a presentation and design tool, a communication design tool, a performance modeling tool and a component modeling tool.

27. The method of claim 25, wherein said reverse engineering tool may be selected from the group consisting of a system structure analysis tool, an extraction tool, a repository population tool and a restructuring tool.

28. The method of claim 25, wherein said construction tool may be selected from the group consisting of a publishing and page mark-up tool, a source code editor tool, a generation tool, a compiler/liker/interpreter/debugger tool and a construction utility tool.

29. The method of claim 25, wherein said configuration management tool include a version control tool and a migration control tool.

30. The method of claim 24, wherein said environment management tools may be selected from the group consisting of a service management tool, a system management tool, a managing change tool and a service planning tool.

31. The method of claim 24, wherein said program and project management tool may be selected from the group consisting of a planning tool, a scheduling tool, a tracking tool and a reporting tool.

32. The method of claim 24, wherein said team productivity tool may be selected from the group consisting of a E-mail tool, a teamware tool, a publishing tool, a group calendar tool and a methodology browsing tool.

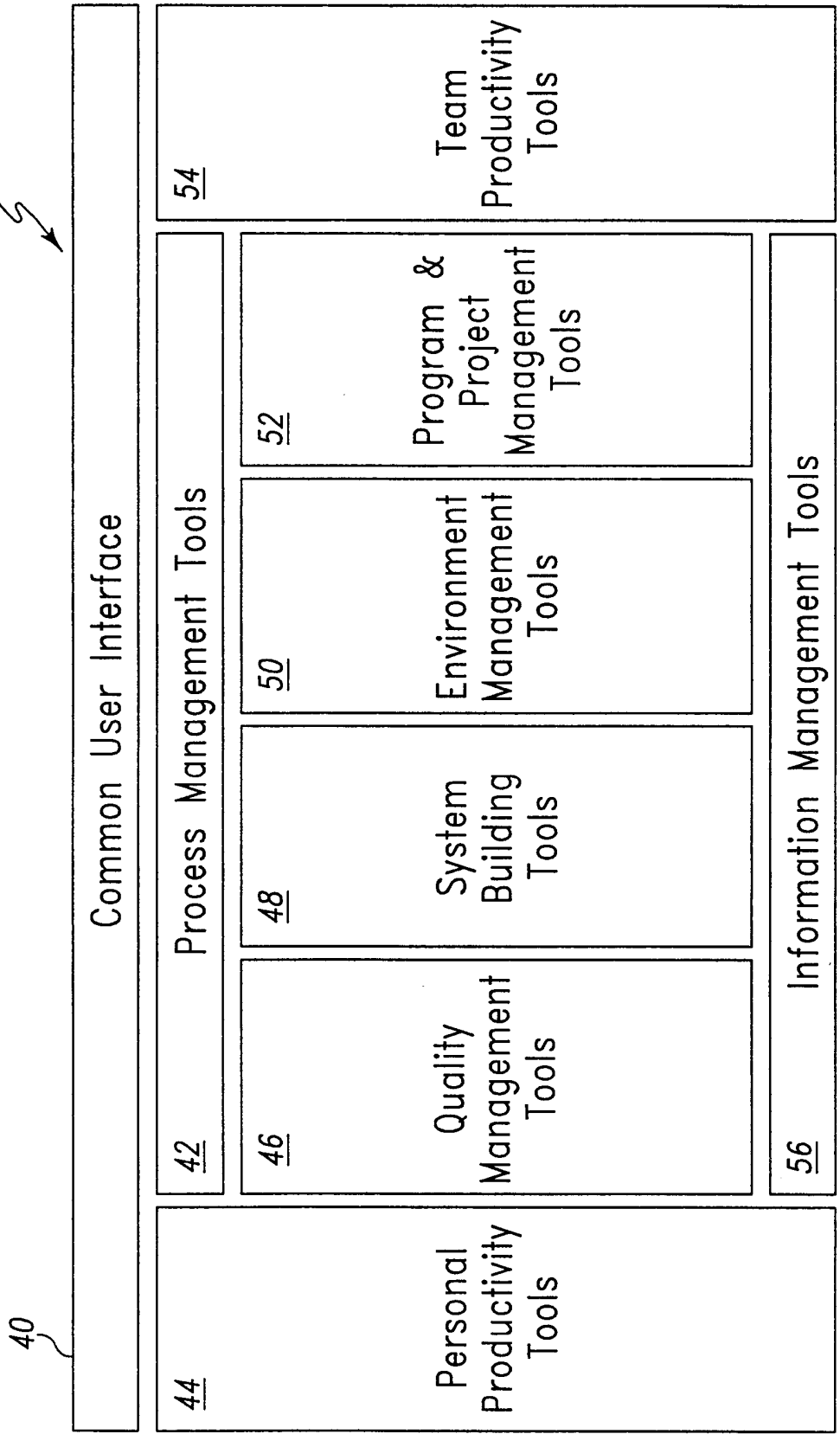

10
40
Common User Interface
44
Personal Productivity Tools
42
Process Management Tools
46
Quality Management Tools
48
System Building Tools
50
Environment Management Tools
52
Program & Project Management Tools
54
Team Productivity Tools
56
Information Management Tools

Fig. 1

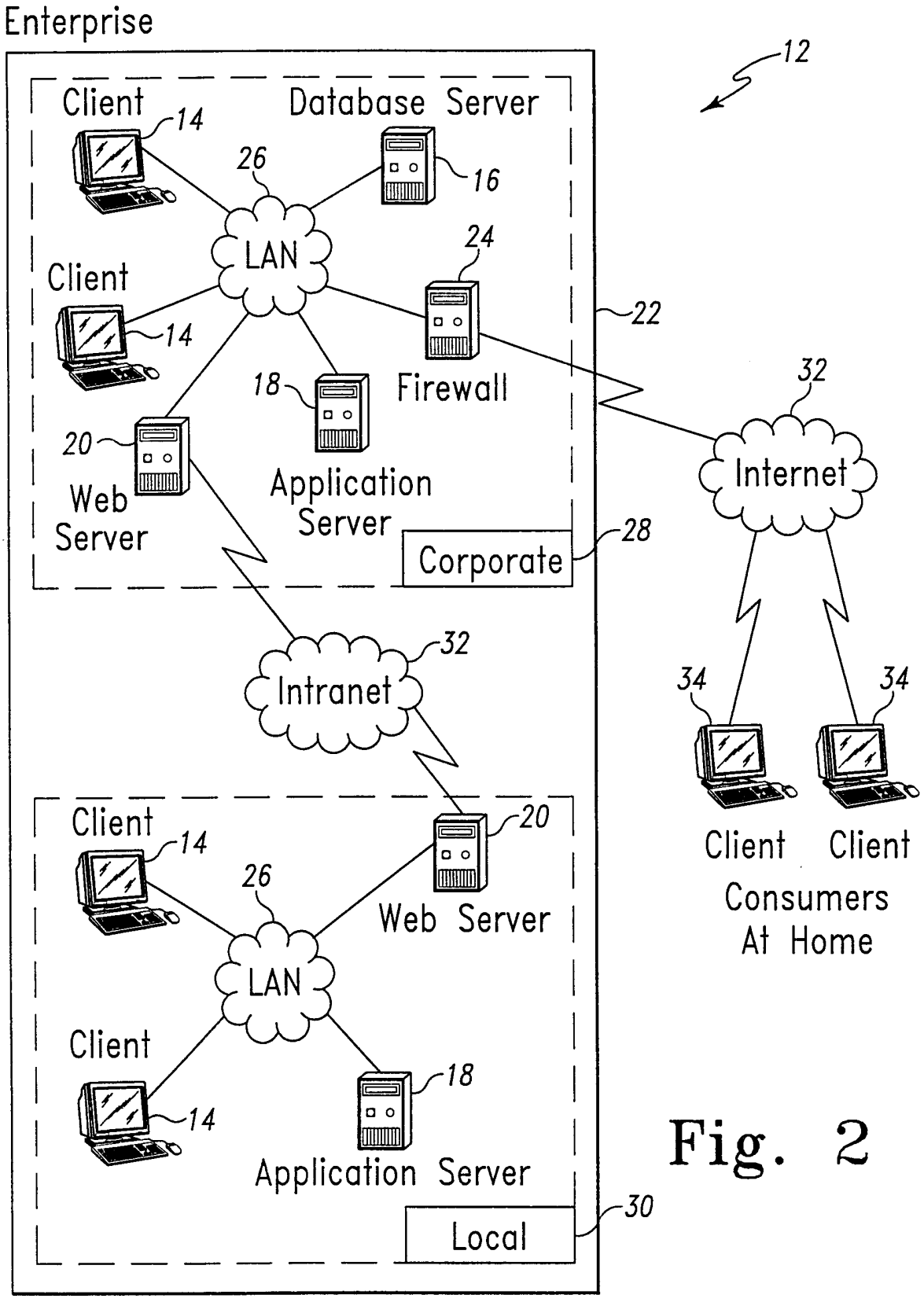

Enterprise
12
Client
14
Database Server
26
16
LAN
24
Client
14
22
18
Firewall
32
20
Internet
Web
Server
Application
Server
Corporate
28
32
Intranet
34
34
Client
Client
Client
Consumers
At Home
20
Client
14
26
Web Server
LAN
Client
18
14
Application Server
30
Local

Fig. 2

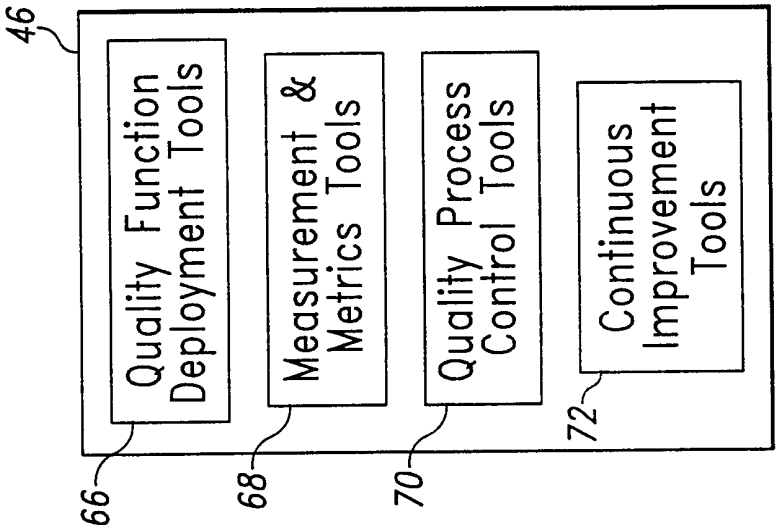

46
Quality Function Deployment Tools
Measurement & Metrics Tools
Quality Process Control Tools
Continuous Improvement Tools
66
68
70
72

Fig. 4

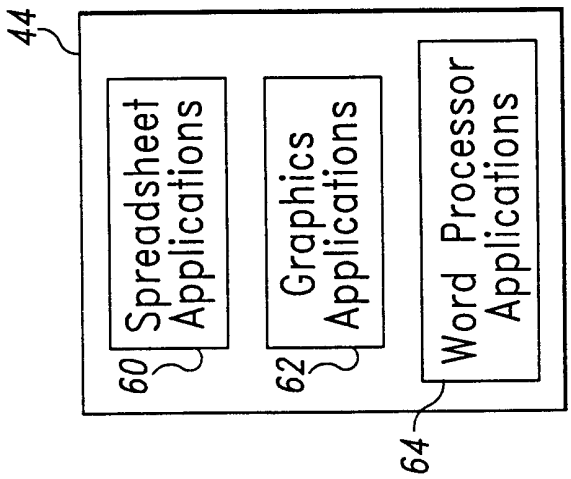

44
Spreadsheet Applications
Graphics Applications
Word Processor Applications
60
62
64

Fig. 3

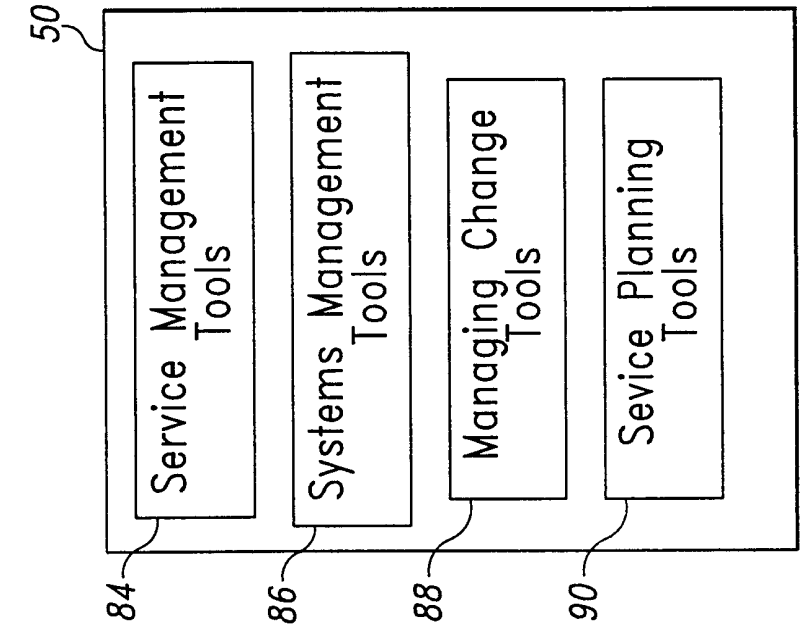

50
Service Management Tools
Systems Management Tools
Managing Change Tools
Sevice Planning Tools
84
86
88
90

Fig. 6

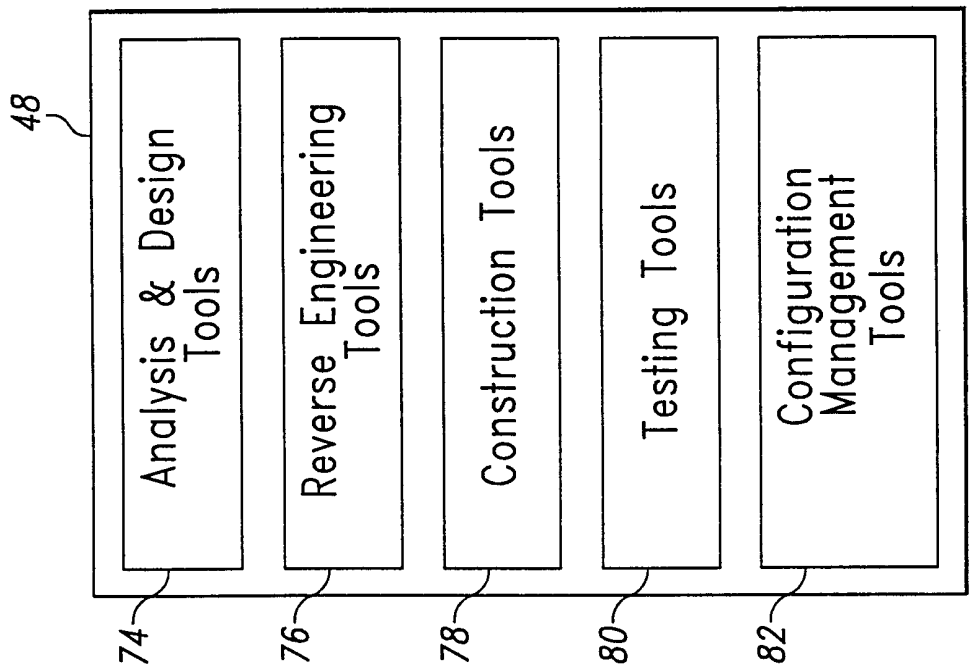

48
Analysis & Design Tools
Reverse Engineering Tools
Construction Tools
Testing Tools
Configuration Management Tools
74
76
78
80
82

Fig. 5

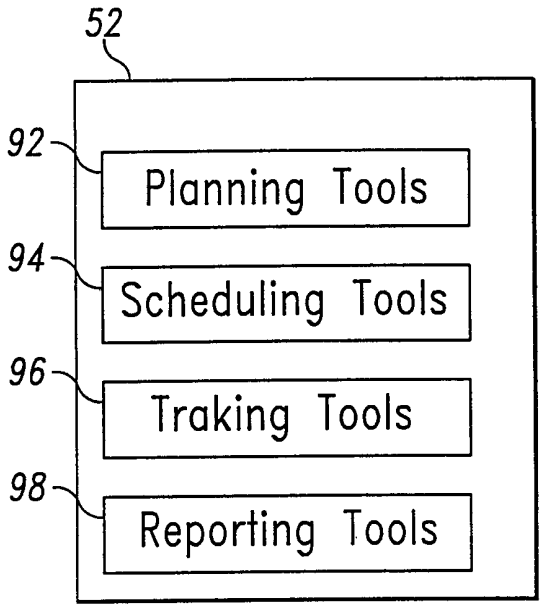


Fig. 7

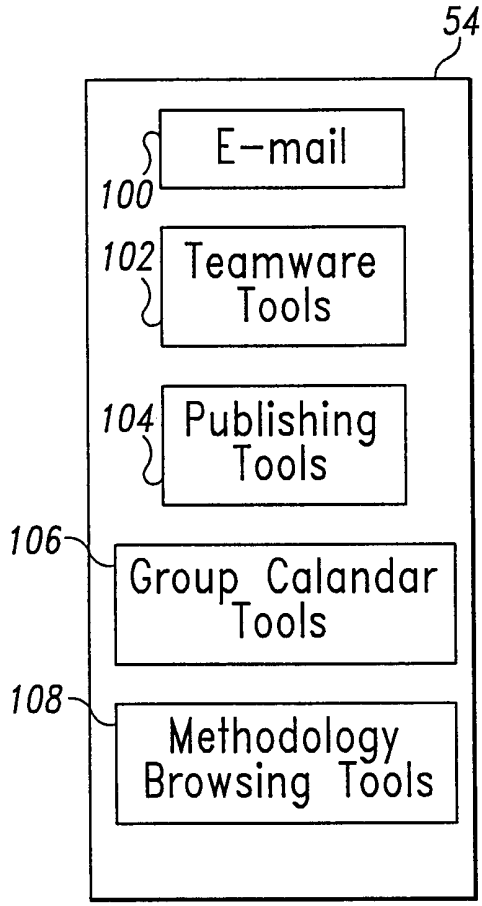


Fig. 8

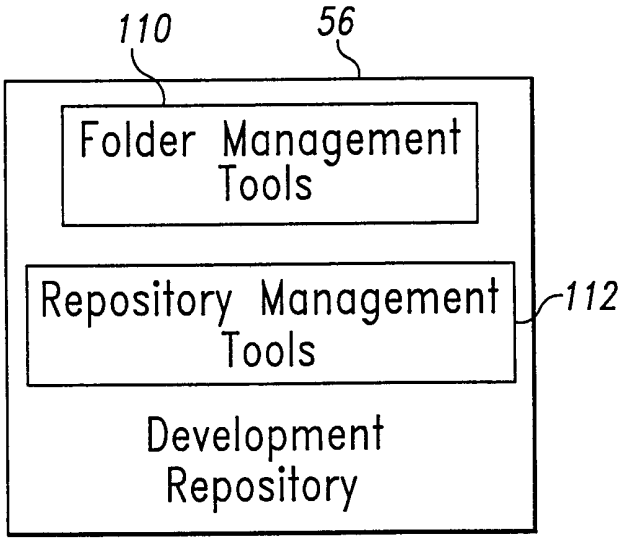


Fig. 9