

update, delete, select by relation, and update fields. These standard operations are intended to be easy to understand and use. The usage of these standard operations is understood through the strict rule set of the meta model of the repository 18. Furthermore, the repository 18 also includes documented modeling of the side effects of the usage of the operations. The side effects for an operation model which stored business objects are affected by executing the method. For example, "delete" usually has a side effect on other stored business objects related to the deleted object. Other standard operations perform more specialized tasks and support functionality for transactions between front end application program 12 and service manager 16 (e.g., a lock operation).

[0034] The canonical interface provided by the service manager 16 also includes specialized actions that are defined for specific classes of business objects and queries that can be defined for clusters of classes of business objects. The clusters are modeled as service modules (described in more detail below) in the meta data. These actions and queries are also defined in the meta data of the repository 18.

[0035] During execution, front end application program 12 issues service requests to service manager 16, service manager 16 checks the requests for consistency with the meta data in repository 18, and then the service manager 16 passes the requests to back end service providers 26 according to the meta data in the repository database 18. The manner of implementing the set of back end service providers 26 and data in database 24 is independent of application 12, with back end service providers 26 and data in database 24 conforming to the definitions and descriptions of the meta data in the repository 18. Database 24 can be a relational database. However, database 24 can be modified to use a different mode of data organization other than a relational database and front end application program 12 does not need to be modified if back end service providers 26 and data in database 24 still conform to the meta data in the repository 18. One such different mode of data organization for database 24 can be an object-oriented database.

[0036] Front end application program 12 provides user interfaces displayed on monitor 10. Front end application program 12 provides application code to display and aggregate the data received from the set of backend service providers 26. Front end application program 12 generates requests, via service manager 16, to the set of backend service providers 26 for standard operations such as select, insert, update, delete, and execute in addition to more specialized operations. Front end application program 12 is interaction-centric, focused on aggregating data of the back end service providers 26 and combining interactive steps into a flow of screens and syndicated screen elements.

[0037] Front end application program 12 contains screen-flow logic of User Interface (UI) oriented applications and front end application program 12 binds a UI to the meta data in repository 18. Front end application program 12 can be indirectly bound to a specific set of backend services by back end service providers 26 via descriptions of the services in the metadata of the repository 18. Front end application program 12 can also be formed from various generic interaction-centric front-end layers that are only bound by configuration to a highly standardized service layer by service manager 16 serving as an intermediary to back end service providers 26.

[0038] In some implementations, a service manager proxy 14 gives the front end application program 12 a buffered access to a service interface provided by service manager 16. Service manager proxy 14 is a server on computer 4 that acts as an intermediary between the front end application program 12 and the service manager 16 so that the business software architecture 2 can ensure security, administrative control, and caching service. The service manager 16 offers a queuing functionality, which is used by the front end application program 12 to bundle several service requests or commands (resulting in service methods) into a single service method queue in order to save round trips. Service manager proxy 14 allows front end application program 12 and service manager 16 to be separated onto different computers 4, 6. Furthermore, use of service manager proxy 14 can allow service manager 16 and the set of backend service providers 26 to be distributed over multiple computers.

[0039] In one example, the service manager proxy 14 communicates with service manager 16 using SOAP (Simple Object Access Protocol) messages via network 20. SOAP is a way for a program running in one kind of operating system (such as a Windows® XP Operating system available from Microsoft Corporation of Redmond, Wash.) to communicate with a program in the same or another kind of an operating system (such as Linux) by using the World Wide Web's Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML) as mechanisms for information exchange. Since Web protocols are installed and available for use by all major operating system platforms, HTTP and XML provide a solution to a problem of how programs running under different operating systems in a network can communicate with each other. SOAP specifies exactly how to encode an HTTP header and an XML file so that a program in one computer can call and pass information to a program in another computer. SOAP also specifies how the called program can return a response.

[0040] As shown in FIG. 3, the service manager 16 provides an interface (defined by the meta data in repository 18) to front end application program 12 that hides the characteristics of the corresponding back end service providers from the set of backend service providers 26 and data in database 24. Front end application 12 uses this interface to retrieve data from backend database 24 to display in graphical user interface (GUI) 28 for interaction with a user.

[0041] The service manager 16 provides the interface to front end application program 12 by receiving and executing requests from front end application program 12 to backend service providers 26. After each receipt of a request by the service manager 16, the service manager 16 delegates the request to one or more service providers 30, 32, 34, 40, 42, 44, and 46. Service provider 30 is an instance of a software class repository service provider. Service providers 32, 34, 40, 42, 44, and 46 represent instances of software classes such as query service provider class (32), aspect service provider class (34), transaction service provider class (40), locking service provider class (42), action service provider class (44), and query relation class (46). The software classes for service providers 32, 34, 40, 42, 44, and 46 can be implemented as ABAP global classes maintained by the ABAP class library using the ABAP development environment available from SAP of Walldorf, Ger-