

priate object. A factory may use the request to construct a new object each time. A pool may return objects that it holds, and accept the object back into the pool after it is no longer in use by the process. The components available for selection can be configured to allow model substitution, object substitution and functional substitution at run-time. By creating side-effect free strategies, fewer models need to be developed due to small differences in functionality.

[0092] In embodiments of the meta-implementation layer and component integration engine of the present invention, a set of persistence engines is responsible for managing objects that describe common data structures (such as customer, address, inventory, images, etc.). These persistence engines are side effect free. Programs accessing data through a persistence engine have no way of knowing what specific object implementation will be retrieved from a manager, only that it will fit the type of object requested to perform a specific task. This decouples object connections, since objects reference the persistence engines or managers to access other objects rather than referencing these objects directly. This indirection allows model substitution, object substitution and data substitution to occur at run-time. Any data for which metadata can be retrieved can be managed by a persistence engine. By using persistence engines for data, fewer models need to be developed due to small differences in data structure.

[0093] The present invention also provides a method for configuring and accessing any object or structured data in the running computer environment using metadata. Objects for retrieving metadata, called "customizers", are managed by the customizer manager. An appropriate customizer can be configured for every structure (object or data format) in the system. For any object for which a customizer has not been setup, the system defaults to a customizer that uses dynamic discovery techniques to retrieve metadata. The ubiquitous metadata allows programs to always externalize configuration, input data, and instructions, allowing programs to be extremely flexible and manageable.

[0094] A command architecture of the present invention uses one or more commands which perform small tasks working together to perform more complex tasks. Metadata exists to describe input, instructions, and output. The metadata allows any metadata-aware component to use any command without requiring it to understand different interfaces for each subject area. In this way any commands can be combined together without requiring code to match different interfaces for each new subject area. Any information necessary for using a command is described by the command's metadata, and understanding of the subject area is left to the programmer or computer program that assembles the commands into a program. This singular way of dealing with all commands eliminates the proliferation of component integration code.

[0095] The component integration engine of the present invention may use centralized managed resources. Managed objects may be used without side effects to the managed resources or to independent processes. Mechanisms exist which can be used to protect the shared resources from alteration after registration in the manager. Additionally, shared resources are not allowed to change themselves or any other shared resource except locally within the process using the shared resource. Many processes can use a single

shared resource and anything occurring within one process will have absolutely no affect on any other process.

[0096] The centralized managers are themselves managed by a single manager, the "ManagerManager", which all other components can access in order to list any manager and any managed component. Managed objects are stored by identity (which is not necessarily a string name as is required by naming contexts), which allows components to be found by any component that knows the identity. Managers may also expose a query interface to allow components to be discovered by characteristics of the component, even when an identity is unknown. Managers may restrict the type of component allowed to be registered or may allow dynamic registration of components. The ManagerManager allows dynamic registration of managers, which allows any manager to be replaced by a different type of manager to allow for different performance characteristic in the centralized management of resources. Examples include "local in-memory manager", "database managed manager", and "LDAP managed manager."

[0097] Managers available on the network allow many component integration engines to share a common set of managed resources. Component integration engines can share these same managed resources regardless of programming language, operating system, database, file structure or any other technical specification aside from common network access. Metadata for component customization of all objects and structured data in the running system may be understood by humans or computers.

[0098] Compiled configuration objects, called "customizers", are a managed resource. By selecting a different customizer from the customizer manager, different mechanisms can be employed to configure the same type of object. This allows for dynamically changing the configuration mechanism, but offers the speed of unchangeable, compiled code. In other words, since the metadata and customizers are part of the system, they are also customizable. A configuration is not limited to attributes; configuration of an object may involve calling specific constructors, setting attributes, invoking methods and registering to receive specific events. Data used in the configuration process is always external to the source code, never compiled into the code. This allows administrators to change the functionality of any component at run-time or replace a specific component with a different component at run-time, completely changing a system's performance characteristics. A developer can add or change components in a running system to implement new functionality or change existing functionality without recompiling or even stopping the running system.

[0099] Metadata is not limited to the object-oriented structure but can be applied to any structured data. Object metadata is a specific implementation related to accessing object-oriented data structures composed of constructors, data, and methods. Object metadata is specific implementation of metadata as applied to objects.

[0100] Metadata-aware objects can access hierarchical data or relational data through the metadata as if it were an object. Recognizing the larger pattern of metadata describing any structured data significantly expands the power of a component integration engine by allowing transparent access to non-object structured data as if it were an object.

[0101] Since metadata does not require objects or data to implement a specific interface or naming convention, any