

[0401] The tool provides meta-implementations for many existing APIs provided as part of a programming language and meta-implementations for accessing database structures as models. The persistence engine and metamodel attributes ensure proper storage of business data. The tool may also be open to third-party metamodels and meta-implementations.

[0402] The virtual operations and meta-implementations are combined via metadata to perform new operation implementations for virtually any type of application. These operations are similar to those in a programming language: Boolean and numeric expressions, branching statements, and loop statements. These logical operations are combined with the common services (database access, email access, image operations, etc.) provided by the meta-implementation provider to assemble complex operations in the same way programming languages create operations out of existing libraries of application programming interfaces (APIs). In fact, by creating a meta-implementation descriptor for the existing APIs in a programming language, all of the functionality of a programming language can be made available as meta-implementation operations.

[0403] A component integration engine is capable of all of these functions. It serves to integrate components through descriptions of the components and descriptions of processes. Rather than generating source code to implement each metamodel accessor in a specific way, the component integration engine uses assemblies of accessors to create software applications.

[0404] Use of a component integration engine to build software using a meta-implementation layer dramatically decreases the amount of time necessary to implement a model. Rather than first modeling the software, then building that software; the modeling process is also the building process. Modeling is required to be a complete, but the modeling process may always have been complete.

[0405] While some metamodel components require special programming to complete an implementation, the meta-implementation provider can provide a wide range of operations as a set of common services. These services implement commonly used operations in computer software like object-relational mapping, object-XML mapping, object instantiation and configuration, naming and directory operations, database operations, email operations, DNS operations, logging operations, object caching and pooling, web accessibility, image generation, text formatting, batch scheduling, and resource security. Since the meta-implementation layer provides these services, they can simply be defined as part of the metamodel without requiring additional implementation.

[0406] Meta-implementation layers are extensible. The meta-implementation layer is extensible since any object wishing to participate in the meta-implementation layer can simply implement the appropriate interface and begin participation. Being extensible means new model concepts and technology components can be added directly to any component integration engine implementation.

[0407] Applications built using meta-implementation are easier to maintain. The meta-implementation layer imposes loose coupling between components in an application. Each component accesses the other components through a very limited number of interfaces (the meta-implementation layer

interfaces), no direct access occurs. Therefore, changes to the implementation of one model require a minimal or no changes to other models. Models are isolated from implementation errors in other models by the meta-implementation layer. Operations that comply with the metamodel definition for preconditions, postconditions, parameter value constraints, parameter access constraints, service access constraints, and metamodels access constraints are unlikely to propagate errors undetected through the meta-implementation layer. Care must still be taken to thoroughly document side effects of operations since the component integration engine may not check for all possible side effects. Side effects are effects resulting from executing an operation that are not obvious through changes to the parameters or return type. An example of a side effect is running a findSquareRoot(int) operation to retrieve the square root of a number, and finding out that this program has entered data into a database table. While this functionality might be correct, it is a side effect, since it is not obvious that some external change has been made.

[0408] The component integration engine of the present invention reduces complexity by simplifying or eliminating the mapping of a metamodel to a model to an implementation. It reduces time to implementation by automating the implementation from the metamodel. It reduces software code maintenance by not producing code and by providing common services. It reduces maintenance by providing easy to change, loosely coupled assemblies. It allows for the integration of any new or existing technology through meta-implementation.

[0409] Domains in the Component Integration Engine (CIE) are the logical grouping of resources, meta-implementations, services, and a persistence engine as they relate to a specific subject matter or problem domain. None of the components available in a domain can be accessed without the user of the domain first being authenticated.

[0410] Authentication is the process of confirming a user's true identity. Authentication in a component integration engine is delegated to an authentication component. Like all other components in a component integration engine, the authentication component is configurable and swappable. This "pluggable" approach to authentication allows any valid authentication component to perform the authentication according to the current configuration. Components for authentication can then be developed for biometrics, UNIX login, database login, Windows NT login, file-based login, Kerberos login, smart card login or any other current or future authentication technology.

[0411] The problem domain holds resource managers to manage shared resources. Shared resources are pre-configured component instances that provide details about the running environment or access to resources, services, or functionality that can be shared amongst all users and processes running in the domain. Example resources are database connections, thread pools, object caches, and resource stores. One manager exists for each type of shared resource. Managers are accessed by the name of the resource they manage. Resources are retrieved from the managers by name or query.

[0412] Package accessors are described above. The package accessors in a component integration engine allow the CIE or any program using the CIE the ability to access any