COGILITY® SOFTWARE

FEATURES LIST
COGILITY STUDIO 6.0

Cogility Studio
Integration Solutions Designed to Work
The Way You Think

Cogility Studio™ is a tightly integrated, robust software environment for building and deploying mission-critical corporate applications.

Cogility utilizes a Model-Driven approach where much of the technical plumbing needed to develop enterprise applications is automatically achieved, leaving the user to focus more on developing core business processes and functionality. Resulting applications are developed faster than with traditional code-centric approaches. They are malleable and more resilient to change.

Enterprise Application Integration (through web services orchestration, messaging and integration of external databases) and complementary business functionality come together as a Composite Application that is described in one consistent and holistic model. The application is then automatically deployed by a single button push. The "pushed model" directly executes without any additional manual translations. The execution platform is made up of J2EE application servers and relational databases.

Cogility employs an object-oriented approach to modeling that is based on current standards like J2EE, JDBC, SOAP, JMS and XML. However, the modeler is shielded from the low-level programming usually needed to interface with these technologies.

The following is a partial list of some of the functions that are automatically managed by Cogility

- Mapping business objects (UML classes and associations) to relational databases.
- Automated persistence of business data without requiring low-level JDBC programming.
- High-level modeling interfaces for in/outbound Web services without requiring SOAP level programming.
- Modeling interfaces for in/outbound messaging without requiring JMS level programming.
- Modeling interfaces for XML manipulation without requiring XML level parsing, such as SAX or DOM.
- Automated deployment of the application to the app-server/database platform.

Cogility Studio is modular in its design and is comprised of three components:

Cogility Modeler

Cogility Modeler provides the application developer with a comprehensive, composite application modeling environment. Modeling changes are automatically saved to a persistent repository, to prevent accidental data loss. Cogility also provides fine grained hierarchical Configuration Management, allowing multiple modelers to effectively collaborate on the development of applications.

Cogility Manager

Cogility Manager is the runtime piece that physically executes inside the application server. It is responsible for executing all of the logic defined in the model.

Cogility Insight

Cogility Insight provides a number of tools for reporting and analysis within the composite application environment. Besides business object data, Cogility also transactionally persists the status of long-lived business processes. This makes it possible for Insight to report on live system data as well as the state of individual business processes.

Modeling

Business Data

- Structural aspects are visually modeled using UML Classes and Associations
- Automatic mapping to a Relational Database
- Easy-to-use Action Semantics for data access, without the need for low-level database access programming

Business Processes

- Business Processes are visually represented with UML State Machines
- Long-lived, asynchronous business processes can be modeled. These processes are persistently managed so they survive system failures. Often, multiple app-servers will be employed for additional scalability. It is important that any app-server can process a segment of a particular process, at different points in time.
- Process logic is partitioned on object-oriented principles across multiple business object classes as appropriate.
- Event-driven enterprise. Cogility provides a layered modeling scheme where often changing details of integration with external systems and databases are described separately from the core business processes. This facilitates a stable description and enforcement, of desired business functions across a changing mix of systems being integrated.

Action Semantics

- Action Semantics are used to affect different parts of the model with specific logic. For example, at some point during an order fulfillment process, it may be necessary to retrieve the customer's shipping address and compute the delivery date based on the Zip code.
- Compile and Consistency checks
    - Action Semantics are statically checked against the model and errors and inconsistencies are flagged.
    - The holistic nature of the model allows various higher level semantic checks and controls that would not ordinarily be possible for a code-based solution.
- Integration with external Java libraries. Often times, third-party Java libraries will be available that do useful functions. For example, an API to encrypt sensitive business data. Rather than require this functionality to be recreated or recoded inside the model, Cogility allows these APIs to be pulled into the model and used as part of Action Semantics. The integration is seamless; a set of Action Semantics references and manipulates both internal model artifacts as well as external Java APIs. Consistency check and compile failures are reported uniformly against all Action Semantics.
- Model and syntax assist is available during editing. This is equivalent to code-assist available in Java development tools.
- Action Semantics are displayed in color to enhance the user experience and the color schemes are user customizable.

Web Services

- Web services can be modeled at a higher business level, without the need for low-level programming
- Modeled Web services are automatically deployed to the J2EE environment upon push. This simplifies and speeds up the development process.
- Outbound calls to Web services defined in other system are similarly facilitated by automatically converting their WSDL to model artifacts in Cogility
- Built-in support for manipulating XML. Lower level programming for XML parsing is not needed. Once the structure of the XSD schema is exposed to Cogility, XML manipulation is easy and syntactically identical to manipulating business objects.
- Both SOAP as well as HTTP web services are supported. While SOAP web services can be employed for traditional B2B integrations, HTTP web services provide a lighter weight alternative for things such as Corporate Intranet clients, as well as tighter integration with AJAX technologies.

Scheduling

- Cogility's scheduling function allows repeating and ad-hoc tasks to be scheduled from within the model, thereby allowing yet another business function to be operated and controlled from one place. The task scheduling and the scheduled activity details are both described in the same model and Operating System level schedulers need not be utilized.
- Tasks can be scheduled in a structured fashion through Cogility Insight's user interface. Tasks can also be dynamically scheduled as appropriate, by any logic execution inside the Model, providing additional versatility.

Messaging

- Since Cogility runs on J2EE application servers, messaging is accomplished through the Java Messaging Service.
- Just like Web services or database access, messaging is accomplished in the model at a higher level. The modeler does not have to delve into the low-level JMS API in order to publish or consume a message.

Domain Transformation Modeling

- Often, there is a need to integrate multiple business systems with disjoint and partially overlapping data models.
- Cogility allows explicit modeling of and transformations between, the data models of multiple systems, based on the CWM specification from OMG.
- Cogility employs a hub-and-spoke model to reduce direct coupling between different data models and eliminate the brittleness that results from point-to-point interconnections.
- The hub or the central Data Transformation model is contained in the same Business objects that implement the common Process model. This allows the enforcement of a uniform view of the business that is flexible and resilient to changes.

Operations

- A set of Action Semantic commands can be grouped as an Operation on a business object. Unlike the long-lived, asynchronous Business Processes, Operations are chunks of synchronously executing commands.
- Operations can be partitioned along object-oriented principles across multiple business object classes. Operations allow Action Semantics reuse.
- An Operation, as defined in the UML specification, is implemented as a pair of artifacts viz. a 'definition', which defines an interface or signature to its callers, and a 'method', which contains the implementation. Inheritance and polymorphism are supported.

External Databases

- Besides connecting to systems via messaging or Web services, Cogility allows external databases to be directly accessed from the Model.
- Java or JDBC level programming is not required.
- Two modes of access are supported:
  - Access of a database is initiated by a business process or Web service executing in Cogility.
  - A change to the database contents initiates access to logic defined in the model.

Statistical Chart Views and HTML-based Graphical Reports

- Can be modeled to view and visualize the contents of the database. Chart Views can be created as:

Pie Charts	Line Charts
Point Charts	Bar Charts
Area Charts	

- Chart views can be linked together to drill down from higher level charts to more detailed charts.
- HTML views and their associated objects define HTML web applications deployed by Cogility Modeler to the application server. When the model is deployed to execution, the web-based views are automatically available.
- Standard Cogility Action Semantics are used to populate the contents of these views.

Development Process Support

Configuration Management

- Fine-grained and hierarchical versioning of model artifacts
- Two and three-way comparisons of model artifacts, coupled with the hierarchical model organization, allows multiple people to collaborate on a model and easily merge their work.
- Support for export/import of entire model as a single file. Works well for small and medium sized projects because a single file can be easily tracked and managed.
- Support for fine-grained multiple files/directories format. Works well for very large models because import/exports are fast. Also, off-the-shelf CM tools can be leveraged, if desired, for handling the ASCII text files.

Model Push and Execution

Deployment Process

- **Easy**
  Model changes are directly deployed to the running application server and database with the push of a button. There are no additional manual steps involved.
- **Metadata-based**
  The model is converted into metadata that resides in the execution database. This means no code-generation and fewer artifacts that need to be deployed to the application server.
- **Fast Delta Pushes**
  Metadata-based execution allows the contents of the Model to be compared to what is currently in the execution database and only changed artifacts are pushed to the database.
- **Fast**
  No application or app-server restart. Since the metadata resides in the database, in most cases, no new artifacts need to be physically pushed to the application server. This means that the application does not have to be shut down during the push process. At the end of the push, all metadata changes are pushed to the database and committed. At this point, the application server's connection to the database is refreshed and it starts to execute the newly pushed Model.

- **Flexible**
  Multiple Deployment model artifacts—combined pointer to the app-server and database where the push is targeted—can be created outside of the main Business Model. Users can change the environment that the Model gets pushed to, simply by choosing the appropriate Deployment object, during the Push process. The Deployment object can identify multiple app-server locations, eliminating the need to push multiple times.

Multi-Server Deployment

- Cogility Modeler now supports deployment to multiple application server instances in a single push operation.

Deployment Tool

- The Deployment Tool provides a way to load a Model into a repository and to deploy that Model to one or more AppServers. It processes a script and executes the directives within that script without user interaction (other than selecting the script).

Iterative Deployment

- Iterative development is facilitated by two factors:
  - Model-based rapid prototyping
  - Efficient deployment process described above

Runtime Scalability

- Cogility combines the rapid Model-Driven development process, with the power of a standards-based execution environment. The two components of the execution platform—J2EE app-servers and Relational databases—are widely available, supported and scalable.
- Cogility provides an additional level of scalability by persistently storing process state information in the database and allowing the app-servers to be mere caching/computing devices. A specific step in a long running business process can be handled by any of a number of app-servers that are running the Cogility model.

Push Preview

- Preview the objects that will be pushed to the database before executing the push.

Transactional Semantics

- **Automatic**
  Business logic is automatically wrapped inside transactions. Changes to business objects—made by Action Semantics that are part of a Web service or an individual business process step—are automatically committed to the database as a unit.
- **Process and Business Data Correlation**
  When a particular step in a business process completes, the process status change and any changes to business objects are persisted to the database, as part of one unified transaction. Alternatively, if there is an exceptional condition which causes a particular business process step to become not-completed, then any partial changes made to business objects are also rolled back.

- **Message and Database Transaction Correlation**
  Messages to external systems—that are part of a business process step—are only sent out after business data changes are correctly committed to the database.

Support Tools

The support tools can be used for general purpose testing and debugging by mimicing external systems, allowing the Model to be developed and debugged in isolation.

Webservice Exerciser

- Can import Web service metadata from the execution database.
- Able to invoke modeled Web services that have been deployed to the app-server.
- Import Web service definitions from a WSDL file or URL.
- Web service definitions and data used to invoke them, can be saved and reused.

Action Pad

- Able to read metadata from the execution database and execute snippets of Action Semantics based on that.
- Action Semantics are compile checked, just like they would be if they existed inside the Model. All of the functionality available inside the Model is also available for executing from the Action Pad.
- Can be used to test Action Semantics in isolation—before putting them inside the Model—or to write test scripts to invoke Action Semantics defined in the model and running on the app-server.
- • Action Semantics can be converted into stand alone scripts that can be executed at the Operating System level, either manually othrough a scheduler.

Message Viewer/Editor

- Can be used to publish JMS messages to the app-server
- Can be configured to listen to messages published by the app-server as part of executing the model

Cogility Software
111 N Market St #815
San Jose, CA 94085
(tel) 949.752.4694
(fax) 949.225.4694

www.cogility.com