COMPONENT SUITE

Building Business Applications from Business Components

COMPOSITE SOFTWARE SYSTEMS

BitSlice™

BitSlice Component technology offers pre-fabricated set of components that can be used to provide ready functionality to an application. BitSlice components raise the level of abstraction of system building by allowing the developer to demarcate the application into levels of distinct functionality that could be stitched together to compose the application.

Packaged as a set of readymade business components encapsulating business functionality, these components can be easily plugged into an application thereby saving many lines of code. With the help of these components the software development process gets a head start and business applications can be built quickly and efficiently. Significant savings can be realized in the stages of analysis, design, development and implementation. The application becomes more reliable, robust and of higher quality.

BitSlice components have been used in various domains and have been extended to cater to special application requirements. Productivity improvements and effort reduction of **upto 30%** have been measured in certain project environments !

The **BitSlice** component set can be used for building business applications in areas such as Trading, Manufacturing, Banking, Insurance, Health Care, Accounting, Transportation and many other business domains.

HIGHLIGHTS

BitSlice components address the **Core Business Information** needs of Organizations.

Core Business Information consists of data and processes that are pivotal to all information systems within an organization. The core consists of business entities such as organization structure, operating units, customers, products, selling policies, buying policies, locations, currencies, delivery terms, payment terms, discount schemes and so on. Further, business computations such as calculation of outstanding collections, overdue amounts, calculation of interest on overdue amounts, conversion of amounts to local currency, calculation of currency gain/loss, inventory valuation, lead times for manufacturing, tax calculations form part of core business information processing functionality.

BitSlice-*Core* encapsulates design patterns for processing and maintenance of Core Business information into a set of 'abstract classes' with well defined 'APIs'. The BitSlice-*Core* contains generic components that can be used across a variety of application systems.

The **BitSlice** suite of components extends this core and provides ready-made components for typical business functions such as sales, purchase, order processing, deliveries, payments, customers, products and static information such as countries, states, cities and so on.

The BitSlice Components are in the form of:


Data Selection Lists (e.g., currency selection, discount selection)


Business Lists (e.g., customer business list, product list)


Business Object Components incorporating business logic, validations and computations (e.g., Payment Term, Carrier)


Data maintenance components (e.g. product, tax maintenance)


Database components (e.g., currency table creation, table load)


Higher level 'Component Assemblies' (e.g., payment card, address)


Higher level 'Component Sub-systems' (e.g., currency sub system)

These components can be easily plugged into web pages, application forms / windows to build web based / intranet / desktop applications. The back end business object components can be used to perform critical business calculations and ensure that data is valid and consistent with business rules.

HIGHLIGHTS

BitSlice components provide an infrastructure of business data and functionality for any business system development. They remove the burden of design and development of Core business information modules altogether from the development process thereby providing savings in time and cost while enhancing system quality.

ENTERPRISE SYSTEMS

As enterprise systems grow in size and volume, the 'core business information' have been observed to increase rapidly, and in the process, they get scattered across application systems and go out-of-sync. Over time, the *master data* and its access become inconsistent, inaccurate, and buried in data transaction structures, databases, etc. Inconsistencies arise in the way critical business calculations related to *master data* are carried out across the enterprise systems. Organizations are unable to perform analysis of entities such as customer, vendor or item across all relevant enterprise systems including sales, services, manufacturing and regional enterprise resource planning (ERP).

The lack of a single consistent enterprise mechanism to manage comprehensive core information results in incorrect answers in business processes. The cost of this inconsistency can be enormous for the business enterprise.

BitSlice component set helps reduce this inconsistency by providing a set of well-packaged functionality that ensure that the *master data* remains in sync and is extended as and when required.

MASTER DATA MANAGEMENT

Master Data Management (MDM) provides mechanisms to achieve consistent and comprehensive information across an enterprise. BitSlice business components can provide the MDM infrastructure with a repository that contains metadata about enterprise information, and common business processes and workflows for maintaining the information.

For example, a *payment methods component* would encapsulate various payment methods and their processes, or a *currency component* would encapsulate the data and processes to do currency conversions, etc. This reduces the need for each application to have its own business logic and process to operate on the core data. Moving core business logic and processes out of applications into the infrastructure components corrects the process of creating and maintaining *master data*.

SALIENT FEATURES

- BitSlice components are easily plugged into web pages, application forms/windows to build web based / intranet / desktop applications

- An Online Customizer is provided through which each component can be customized for various features and the customized version may be used in an application or assembly.

- Customization can also be done using the customizer provided by the IDE

- Customizable features includes user interface (look-and-feel) features as well as functional features

- Integrates into IDE for Java such as Eclipse, Sun Studio and IDE for Microsoft platforms such as Visual Basic, Visual C++

- Component set can be extended for Vertical business domains

- All components are provided with easy-to-use high level APIs.
  For e.g., a currency selection list can be created with a single line of code, such as:

  ```
  CurrencyChoiceList c1 = new CurrencyChoiceList();
  ```

  This line of code will create a selection list of currencies obtained from a database of core business information. The list is ready for the user to make a selection of currency.

- Users can pick-and-choose components that they wish to use

- Components can be used as is by embedding them or they may be extended

- Components can be extended both for additional methods and additional data requirements

- Components are available both on Java and Microsoft platforms

- Components available as Java Beans and Class Libraries on java platforms and as COM and Active-X components on Microsoft platforms.

BitSlice-*Core* can be extended to build component sets for *any vertical business domain*.

A complete Customer support Product, called **PowerCare** has been built using BitSlice component set.

BUSINESS OBJECT COMPONENTS

Business Object Components encapsulate business logic and calculations on Core Information. An example is shown below.

```
Net amt = gross amt – discounts –taxes

Payment due date = delivery date + credit days.

Amount in base currency = amount in foreign currency *
exchange rate
```

Business object components provide crucial services like business validations, maintenance of information structure and business computations. For example, the *Payment Term Business Object* does typical computations such as calculation of payment due date on a given sale/purchase, calculation of the status of a sale/purchase whether due/overdue/just due, calculation of number of days after which payment is due/overdue, calculation of interest on overdue payments, etc.

Likewise, the *Discount Business Object* does discount calculations based on typical discount schemes. These non-visual components act as independent agents to provide data and services via APIs to other application units. They form the backbone of business applications and can be used throughout the application systems on various units such as business transactions, reports, online queries, etc. Usage of such components will ensure consistency in business validations, calculations and information structure.

Some of the services provided by Business Object Components are:

- Maintenance functions (Add, Delete, Modify, View)
- Definition and Maintenance of Information Structure
- Business Validations
- Provision for Logical and Physical delete
- Provision for Refreshing the component at any point in time
- Maintaining Date and timestamp of modifications
- Duplicate checking on Names
- Methods for Decoding
- Multi-User operation and Concurrency control

- Maintaining Referential Integrity
- Operations on a single business object and/or obtaining lists of objects
- Overloaded constructors provided so that the object can be instantiated based on Name, Identifier or any other parameters as relevant
- Creation of a Business Object with data retrieved from the database is a single step process through a single line of code, for example:

```
Currency c1 = new Currency(USD);
```

The above statement will retrieve the rates and other details of the currency 'U.S. Dollar' from a database table of currencies and return a Currency object with details filled in. This object may then be used to perform any computations as required.

- Business Computations
  - Standard business computation logic is provided with business components. For example,

  - Some of the computations provided by the Payment Term Object are:

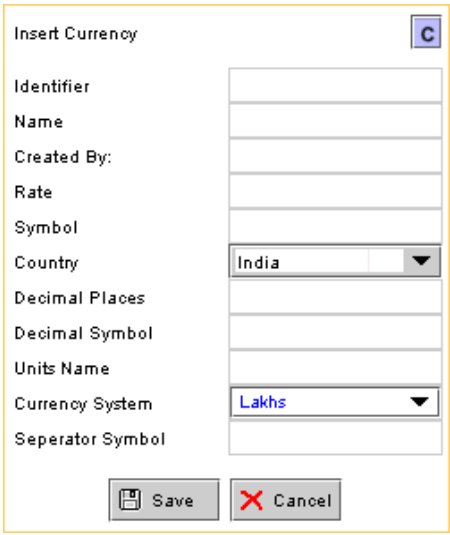
    - Calculation of Payment Due Date on a given Sale/Purchase
    - Calculation of status of a sale or purchase: whether due/not due/overdue
    - Calculation of no. of days after which the payment is due
    - Calculation of no. of days by which payment is overdue
    - Calculation of Interest on overdue amounts
    - Calculation of overdue range in which this amount falls
    - Calculation of due range in which this amount falls

  - Some of the computations provided by the Currency Object are:

    - Computation of currency rate as of a particular date/time
    - Currency rate history
    - Calculation of base currency amount
    - Conversion of amounts between currencies
    - Calculation of currency gain/loss due to a sale amount in given currency
    - Varying number of digits after decimal, varying currency formats
    - Business calculations in given Currency
    - Base Currency setting
    - Truncation / Roundoff methods
    - Conversion of Amounts to words in desired currency

DATA MAINTENANCE COMPONENTS

Data Maintenance components can be used to make changes on Business data consistent with business rules and policies. They make use of the underlying Business Object Component to deliver validated data, perform Maintenance operations, Business Validations and define/maintain the information structure. They can be embedded into any kind of UI container. Data Maintenance component can be plugged into maintenance modules and business transactions. A sample component is shown below.



The following Maintenance activities are supported on all types of Core Business Information:

- Insert
- Modify
- Delete (Logical Delete)
- Undo Delete (undo logical delete)
- Delete (Permanent delete)
- Delete all Logically deleted records

The components can be customized for look-and-feel and functional features.

Creation of a Maintenance component with data retrieved from the database is a single step process through a single line of code, for e.g.:

```
CurrencyModify c1 = new CurrencyModify("USD");
```

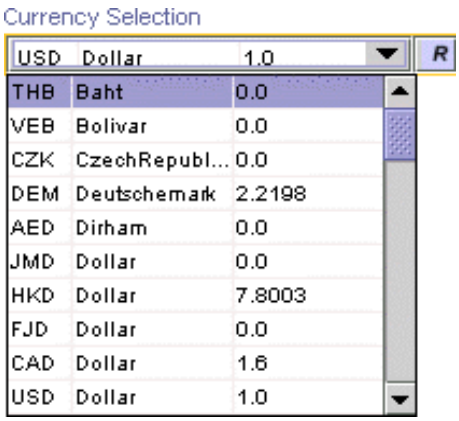
The above statement will retrieve the rates and other details of the currency 'U.S. Dollar' from a database table of currencies and return a Currency Modify object (a UI panel) with details filled in. This object may then be embedded into any kind of UI container allowing for user interactions on it.

DATA SELECTION LISTS

Data selection lists are multi column combo lists packaged with business data retrieved at runtime from a database of 'Core Business Information'. These components can be directly plugged into business transactions such as orders, invoices, payments, cheques, etc in order to make selections on relevant business data. They can be plugged into parameter selection units for reports and queries.

They make use of the underlying Business object Components to provide validated data from the database. They can be embedded into any kind of UI container.

Currency selection list can be plugged into any transaction / report generation unit where currency of sale / purchase has to be selected



All data selection lists are provided with an optional Refresh button that can be used by end-users to refresh the list as and when required. APIs are also provided for refresh which can be used by the container to refresh all contained components.

The lists can be customized for functional features such as:

- Columns that should appear on the list
- The sort order of the list (whether by name or identifier)
- The entry that should be selected by default
- Whether or not the refresh button should appear alongside the list user refresh

Lists can be also be customized for look-and-feel features such as colors, fonts, etc.

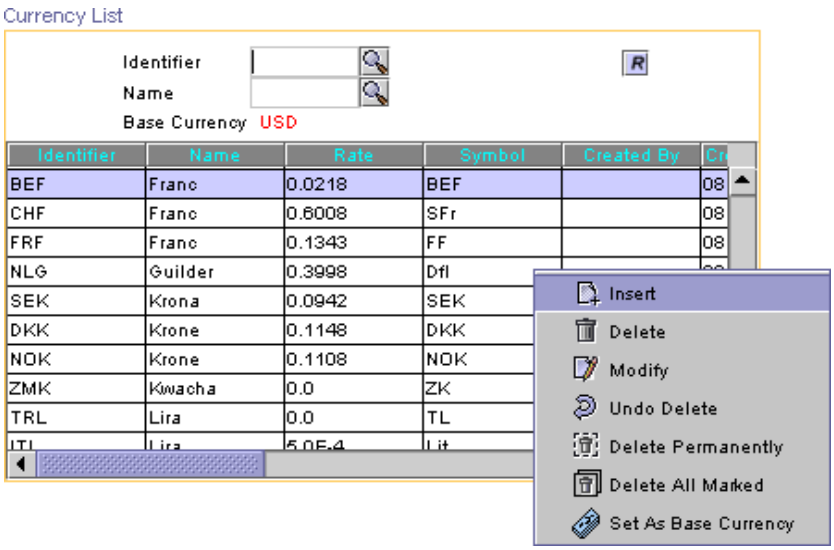
Creation of a selection list component with data retrieved from the database is a single step process through a single line of code, for e.g.:

```
CurrencyChoiceList c1 = new CurrencyChoiceList("USD");
```

The above statement will retrieve a list of currency objects with details from a database table of currencies and return a Currency selection list object (a UI panel) with list of currencies filled in. The selection list may then be embedded into any kind of UI container allowing for user to make a selection on currency.

BUSINESS LISTS

Business Lists provide a tabular view of Business Information retrieved at run-time from the database. They can be used to view and/or operate on Core Business Information such as Credit policies, Currency rates, Delivery terms, discounts, payment terms, etc. in accordance with business validations and rules. They make use of underlying Business Object Components to display and update data. A sample business list is shown below.



All Business lists are provided with an optional Refresh button that can be used by end-users to refresh the list as and when required. APIs are also provided for refresh which can be used by the container to refresh all contained components. Search by Id/Name is available on the list. The list can be made editable and maintenance activities may be carried out on the list.

Business lists can be customized for functional features such as:

- The sort order of the list (whether by name or identifier)
- Whether or not the list is editable
- Whether or not the refresh button should appear alongside the list for user refresh

They can also be customized for various look-and-feel characteristics such as colors, sizes, fonts, graphics, etc.

Creation of a business list component with data retrieved from the database is a single step process through a single line of code, for e.g.:

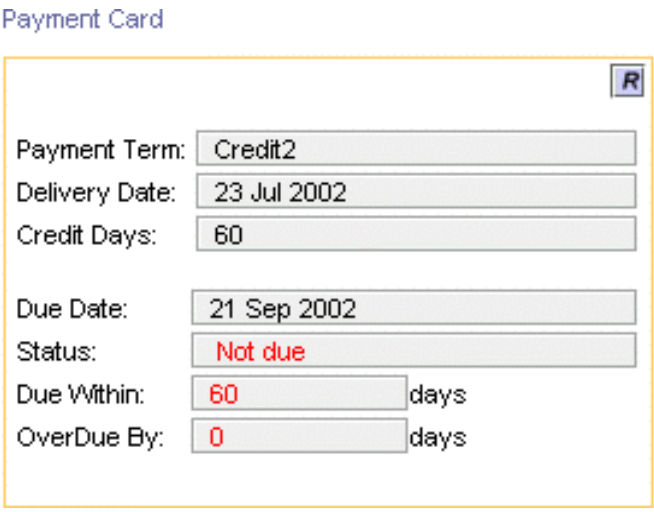
```
CurrencyBusinessList c1 = new CurrencyBusinessList();
```

The above statement will retrieve a list of currency objects with details from a database table of currencies and return a Currency business list object (a UI panel) with list of currencies and their details filled in. The business list may then be embedded into any kind of UI container allowing for user to view or make changes on currency.

COMPONENT ASSEMBLIES

Assemblies are composed from lower level components and are larger in terms of the scope of business functionality offered by them. However, since they are composed from lower level components, they have a flexible structure and can be detached and reassembled according to user needs. Assemblies can be visual / non-visual and make use of other components such as Data selection lists, Business Lists, Business Object Components, Maintenance Components and/or other Assemblies in their composition. They make use of underlying Business Object Component to deliver the data and computations.

The Payment card assembly is capable of calculating and displaying credit and payment details such as payment due date, status of payment, the number of days by which a payment is due/overdue, etc. Customized versions of this Assembly can be used to calculate and display the payment details of Sale or Purchase transaction.



Payment card component shows status of payment on any sale that has taken place.

The Currency card assembly calculates all invoice amounts in local and foreign currency and calculates the currency gain/loss incurred on the business transaction. Customized versions of this Assembly can be used to make currency related calculations on business transactions.

Currency card component shows the invoice amounts in local and invoice currency

COMPONENT ASSEMBLIES

Assemblies like all other components can be customized and embedded into any kind of user interface. Higher level components / assemblies are themselves built from smaller components. Components going into the higher level unit can be individually customized and these can be used to form the higher level unit by providing the name of the customized bean to the higher level unit. This process can be repeated any number of time, making the customization process highly flexible.

DATABASE COMPONENTS

Database components are used to create tables and load business data pertaining to 'Core Business Information' in the application database. BitSlice can be used with any relational database along with the appropriate JDBC driver.

Database components are in the form of:

- Database Connection component. This can be customized to connect to the user's database
- Table creation component
- Table Load component
- Components to update specific data where relevant, for e.g., updating of currency rates in the currency table

The component set comes with sample data where relevant, e.g., Payment methods, Credit cards, Payment terms, Delivery terms, Countries, States, Cities and so on.

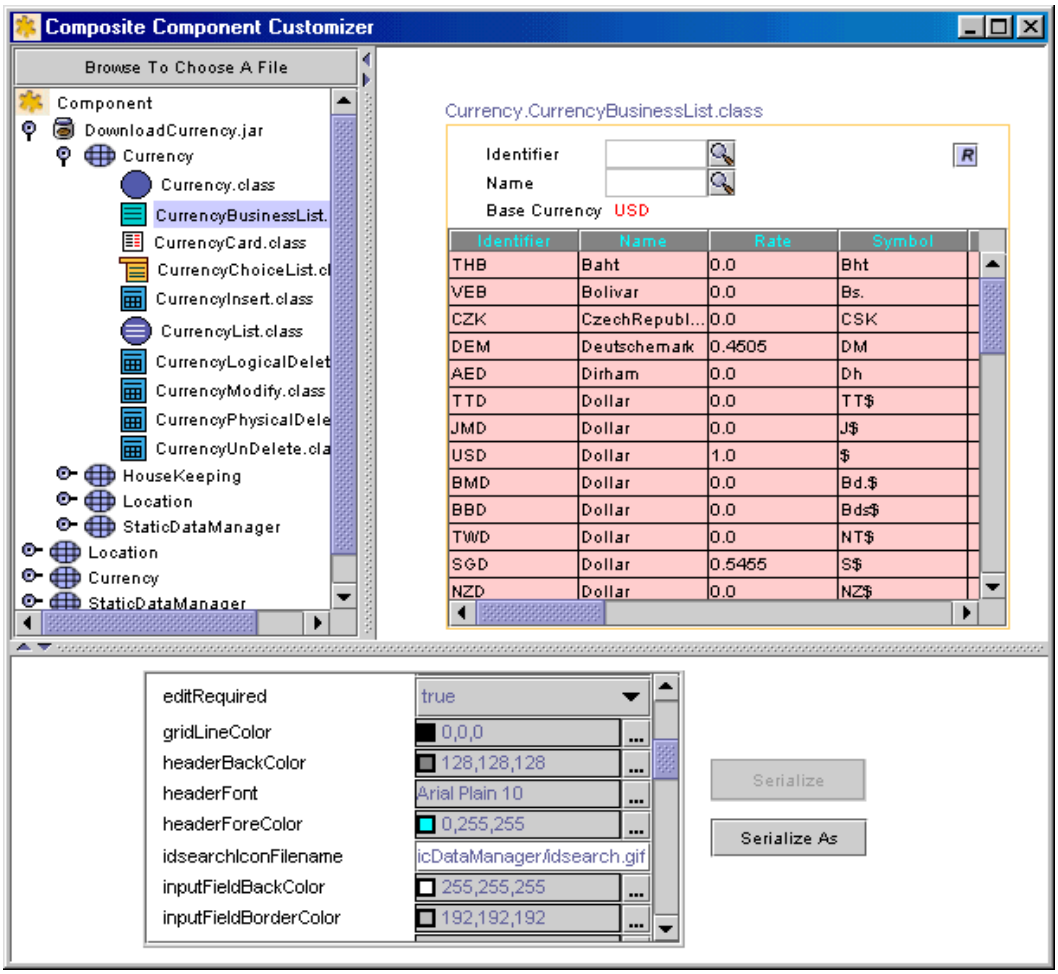
Database components makes use of the underlying Business Object Component to validate data while doing insertions and updates.

COMPONENT SETS FOR VERTICAL SEGMENTS

For any vertical business segment, for example, Insurance, Banking, etc., such a component set consisting of 'Business Objects', 'Data selection lists', 'Business lists', 'Maintenance components', 'Component Assemblies' and 'Database components' may be created for 'Core business information' by extending the BitSlice core abstractions.

The extended component set would provide a uniform service layer for core business information.

All components in BitSlice are customizable by the user. Customization can be done by changing the values of customizable properties associated with a given component. The components may be used as java beans /com components / active-x controls or class libraries. The components can be integrated into any IDE for Java or Microsoft platform and can be customized using the customization mechanism provided by the IDE. Alternatively, BitSlice comes with its own customizer that is a stand-alone java application and this can be used to customize the component. Given below is a snapshot of the BitSlice customizer being used to customize the 'Currency Business List' component:



Customizable properties fall into broadly two categories

- Properties that control the look-and-feel of the component. These are relevant for visual components

- Properties that control the behavior of the component. This is relevant for all components

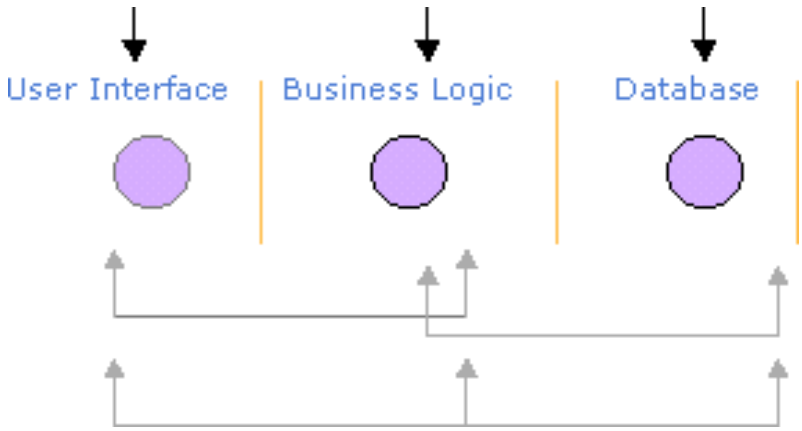
If the components are used as class libraries, they can be customized using the APIs provided. The properties can be set as soon as the class is instantiated in the application.

CUSTOMIZATION

Higher level components / assemblies are themselves built from smaller components. In such cases, the components going into the higher level unit can be individually customized and these can be used to form the higher level unit by providing the name of the customized bean to the higher level unit. This process can be repeated any number of times.

ARCHITECTURE

The application has been partitioned into three typical architectural layers, viz. User Interface, Business Logic and Database layers as shown. In each layer, abstract classes have been built to encapsulate the typical requirements of 'Core Business Information' pertaining to that layer.



For example, the classes BusinessObject and StaticData are abstract classes that provide services related to Business Objects in general. The class StaticDataChoiceList is an abstract class that provides services related to drop-down combo selection lists. The class StaticDataMaintenance and StaticDataInsert are abstract classes that provide services for user insertion of data.

Small, self-contained functional units that are frequently reused both within and across applications have been identified. For each such unit, relevant classes from within and across the architectural layers have been picked out and packaged as ‘abstract components’.

The abstract components have been extended to build concrete business components in generic areas across application domains and specific areas in the Trading and Manufacturing sectors. For e.g., the class 'Currency' is an extension of StaticData class and can be used in any application that deals with multiple currencies. Similarly, the class CurrencyChoiceList is an extension of StaticDataChoiceList and can be used on any transaction / report where a choice needs to be made on Currency.

- Components engineered using O-O technology
- Java language used for development
- 'Abstractions' built for Core Business Information
- Components built by extending the 'Abstractions'

CLASS DESIGN

O-O techniques have been extensively used to build robust high level abstractions for processing of Core Business Information. Thus, the total lines of code written is minimal thereby ensuring high degree of Quality and Maintainability.

Every Business object ‘knows’ and ‘Maintains’ Audit information about how and when it got created and last modified.

Each Object fully validates itself while being constructed with information and before any addition/update to database, irrespective of whether the object was validated by the UI or not. Thus the Business Object layer will ensure that it is updated correctly and can be re-used independently, for e.g., in Export / Import of data.

Constructors are overloaded to construct either an empty Object or construct an Object that is filled with data based on Object identifier. This minimizes the code that the user needs to write.

All Objects provide ‘Refresh’ API to refresh itself with the latest information from the database. The visual objects are also provided with a ‘Refresh’ button that can be used to refresh the UI by the end-user.

Multiple design strategies built in as overloaded/ Additional methods, for e.g., Logical and Physical deletes, Field level and screen level edits, Re-validations for Concurrency control, etc.

Persistent classes are separated out.

Objects validate only data that is owned by it, and not data that comes in as reference from other Objects in the system; In such cases only checks for whether the data is mandatory/optional is done.

ABSTRACTIONS

Design patterns for processing and maintenance of Core Business Information have been encapsulated into a set of ‘abstract classes’ with well defined APIs. Some of the abstract classes are Generic Business Object , Static data Class, Static data list, Business list class and Generic Maintenance classes for Insert, Modify, Delete, Restore and Physical delete.

Generic Business Object has attributes and build in method protocols for handling business objects across various domains. Static Data class has method protocols for operations on ‘Core Business information’. List classes have method protocols for operations on a list of business objects and for constructing and operating on business lists. Maintenance classes have method protocols for maintenance operations on ‘Core Business Information’.

Abstractions standardize the design and build in protocols. The abstract components can easily be extended to build 'component sets' for specific vertical segments.

Higher level assemblies too have been put together from components spanning one or more architectural layers. For e.g., the CurrencyCard, which is a visual unit that displays sales/purchase amounts in local and foreign currency, makes use of components in the user interface layer, business object layer as well as database layer to provide the relevant information on the visual panel.

PLATFORM REQUIREMENT

Java Virtual Machine
Java Plug-in (for visual components in web)
RDBMS, JDBC driver
64MB RAM, 5 MB disk space

CONTACT

Email: composite@compositesoft.com

COMPOSITE SOFTWARE SYSTEMS
email: composite@compositesoft.com.
website: www.compositesoft.com