Agility by Design: Building Software to Last

an eprentise white paper

tel: 407.591.4950 | toll-free: 1.888.943.5363 | web: www.eprentise.com

Author: Helene Abrams
www.eprentise.com

Part 1: A Stable Design

The implementation of enterprise systems brings with it great promise of better information, consistent systems, and reduced operational costs. Achieving that promise, however, is an immense challenge. Building a cohesive enterprise software environment is extremely complex. Most large companies have hundreds of applications to run their business. Many of the systems automate similar business processes and utilize the same data. One of my customers is selling one of their product lines. They have approximately 1700 systems in place. Even if they had an up-to-date inventory of those systems, it is almost impossible to separate the data that is used for the product line that they are selling. They don't know where the information is, and even if they did, all of that data is in different formats.

For all practical purposes, that data housed in 1700 systems is lost to the organization and the value is unrecoverable because of the expense and effort involved in trying to retrieve it. Organizations continuously need to adapt their enterprise systems to keep pace with business change. However, enterprise systems, once deployed, become very rigid, and even small changes are slow and costly to implement. Designing systems to allow for continuous business process improvement, the flexibility to change, and yet provide complete, correct, and consistent information is very challenging and has become the ideal in many organizations. In order to achieve the promise of better systems, agility must be built into the original design. This is the first in a series of articles that discusses how to design, build, implement, and deploy systems that are agile when built and remain so over time.Achieving agility, in both the design and in the resulting implementation, is a multistage process, one having what is aptly described as an Enterprise Application Lifecycle (EAL). Literature in this field generally recognizes EAL as having two phases: the initial implementation and application optimization. The implementation phase consists of activities including defining the strategic direction; analyzing business and technology requirements; obtaining project funding; acquiring hardware, software and integration services; and implementing the enabling solution. The application optimization phase may include adding new functionality or upgrades, application integration, reporting, and data analysis.[i] Traditional descriptions of the EAL process are doubly deficient: (1) they don't address the earlier stages of the process – designing and building applications to last, and (2) they do not describe how these applications mature to provide agility.

Building Blocks of Enterprise Applications

There are three core building blocks of an enterprise application: metadata, data, and business processes. Metadata is the structure of the data, the tables, columns, or objects that contain the data, and the rules governing the insert, update, and delete functions, and control access to the data as it is stored in the metadata containers. In order to be designed for agility, the metadata model must be extensible. That is, it should be relatively easy to add extensions as the requirements change. Designing an extensible metadata model means that the logical design should be fully normalized, that the data dictionary is kept up-to-date, and that all relationships are fully documented with a complete history of changes. There must be an audit path from the logical model to the physical model, and the justification for deviations from the logical model. Naming standards for table and column names along with reusable, documented libraries of procedures make it easier to identify usage of com
mon, shared objects.

Data and their relationships comprise information, and that information is the real competitive advantage of any company. In any enterprise application, there are 4 types of data:

- **Seed data** – data that comes with the application on installation. Examples of seed data include a "standard calendar", or a list of states. Seed data is embedded in the application and generally can't be changed.
- **Configuration data** – parameters for the application that are set up by the user. Configuration data may include a chart of accounts, a supplier's payment terms, or a list of diagnostic codes.
- **Master data** – the families of data that are of interest to the business (customers, suppliers, products, employees, etc.). These are typically the resources of the business and they fit into subject categories. All businesses, regardless of the type of business, store data about their people, products, financial, information, and physical resources.
- **Transaction data** – records the operation of the business processes. Examples of transactions include orders, invoices, or payment records.

Business processes typically go through their own lifecycle phases, especially as they relate to the resources that each process manages. The phases of the process lifecycle are:

- **Planning management of the resource.** For example, planning for the employee resource includes setting up HR policies, compensation plans, and management structures.
- **Acquiring resources needed for the management.** The resources needed to manage the employee resource might include an HR department and recruiting firms.
- **Creating the resource.** For employees, this includes recruiting, interviewing, and hiring decisions.
- **Deploying the resource.** Deploying employees includes assignments (and reassignments) to departments, teams, and tasks and compensating them.
- **Monitoring the resource.** For employees, this might include status reports, evaluations of deliverables, and performance reviews.
- **Retiring the resource.** For employees, this would include accepting resignations, firing, layoffs, exit interviews, outplacement support, and (literally) retirements.

In order to be agile, an enterprise ecosystem – the metadata, the data, and the business processes – needs to be independently sustainable and scalable resulting in a single source of truth. That is, each element of the ecosystem needs to follow the three C's: Consistent, Complete, and Correct. The metadata should start with a single enterprise data model with different subsystems reflecting subsets of the model. To ensure consistency, elements of the enterprise data model must be shared among those subsystems and built from the top down.

Likewise, the process model begins with an enterprise process decomposition with supporting data flows and work flows showing the interactions among the processes. The process model is checked for missing, incomplete, or wrongly modeled processes. Analyzing the process model should show that the processes cover all phases of each resource's life cycle. As a further check on completeness, a data-process matrix identifies which processes Create, Read, Update, or Delete which data elements. This is commonly known as a CRUD matrix. In order to be complete, every data element should be created, have the ability to be read, updated, and deleted, and be used by at least one process, and every process should Create, Read, Update, or Delete at least one data element. Correctness of the models includes checking to see that the representation of the metadata, the data, or the process means the same thing everywhere it is used. There are three main measures of correctness.

- Each object must be represented,
- Each object must be represented only in one place, and
- Each object must only represent one object type.[ii]

For the metadata to be complete, consistent, and correct, the logical data model is fully normalized, each table represents only one class of data, attributes are consistently named and are classified into a domain that represents all of the properties of that domain, (i.e. a phone number column has the same structure whether it is a fax number, a mobile number, or a land line, descriptions are always a standard length), all relationships are complete with cardinality, and the data model contains all objects of interest to the enterprise. For the data to be consistent, complete, and correct, a duplicate resolution process must be applied to all seed data, configuration data, and master data within each application and across the enterprise.

For example, if a unit of measure of dozen is represented in three systems as doz., dz., and dozen in a pull-down list of values, and if those systems are consolidated together, then all representations of dozen would be in the pull-down list of values and the user would be uncertain of which one to pick. This is not as simple as just changing the list of values table since that unit of measure would be reflected in all orders, invoices, purchase orders, and catalogs. If a patient on his first visit to a hospital is noted to be allergic to cillin, and on another visit, that allergy is to amoxicillin, and on the third visit, he said that he has no allergies, is it the same allergy or a different allergy, or perhaps is it a different patient? How should different customer credit limits be determined if they are different in different parts of enterprise systems? Finally, if the data is changed, it should be changed only in one place and then that change should be reflected everywhere in the enterprise. If an address needs to be updated in 27 different systems, then it is difficult to maintain, and if there is a time lag between the updates, it is almost impossible to determine which is the correct address (assuming that the user is able to identify all of the 27 different places where the address is stored). The same principles apply to business processes. A process should only be done in one system, and that process should have defined start and stop points within that same system. If all the inventory, in all the warehouses, are tracked within a single system, it is easy to note how much inventory is on hand, when the inventory item needs to be reordered, whether inventory needs to be moved from one location to another, the value of that inventory, and the most efficient and cost effective means of distributing that inventory to the customer.

When an enterprise reaches the stage of complete, consistent and correct data and standardized processes, then its systems are in a stable state. When the systems are in a stable state, it is possible to consider changes to the data and changes to the business processes allowing the systems to be agile enough to support changing business requirements.

Curious?
For more information, please call **eprentise** at **1.888.943.5363** or visit **www.eprentise.com.**

About eprentise
eprentise provides transformation software products that allow growing companies to make their Oracle® E-Business Suite (EBS) systems agile enough to support changing business requirements, avoid a reimplementation and lower the total cost of ownership of enterprise resource planning (ERP). While enabling real-time access to complete, consistent and correct data across the enterprise, **eprentise** software is able to consolidate multiple production instances, change existing configurations such as charts of accounts and calendars, and merge, split or move sets of books, operating units, legal entities, business groups and inventory organizations.