Functional Capability:
An Incomplete Success

an eprentise white paper

eprentise®

Author: Helene Abrams

www.eprentise.com

How the Implementation of an Enterprise Application Goes from "Finished" to "Done"

Enterprise applications must continually evolve in order to support ongoing business changes. Enabling and facilitating those changes so that they can occur with minimal cost and business disruption is the essence of designing for agility. The backbone of potential design agility is most often the enterprise resource planning (ERP) application, whether it is the Oracle® E-Business Suite, SAP, or another similar system. In larger firms, of course, the ERP systems do not run in isolation – ERP systems are to differing degrees supplemented, complemented, and duplicated by other applications. The average Fortune 1000 company has approximately 300 different "enterprise" systems. Nevertheless, since the ERP systems contain the most central data and business processes of the company, their implementation is a focal point for design agility. Companies do not recognize that a successful implementation of their keystone enterprise application is really much more ambitious than what was envisioned at the start of the implementation process. Many reach a stage at which the enterprise application is up and running and where the old data has been imported and new data can be entered, and they therefore treat the implementation as finished. What they have attained at that point is what might be termed "functional capability" – they have **_finished_** a vital stage of implementation, but they are not **_done_**.

At functional capability, companies are able to conduct business as usual. However, even though the systems work adequately, the new applications often just mimic the legacy systems that they were meant to replace. A system that has only reached functional capability merely supports silos of the business rather than enabling the enterprise as a whole to respond to changes since it does not have common global data standards, common accounting practices, or shared and coordinated business processes. Knowing that the purpose of implementing an enterprise application in the first place is to obtain efficiency gains, **this merely functional system is an incomplete success.**

That is not to say that the functional system will not generate mild improvements. It will capture some gains that generate economies of scale as well as some that flow from increased efficiency in processing data and reduced expenditures for maintaining systems. Nevertheless, the merely functional system leaves the most difficult-to-attain and richest rewards on the table because it is incapable of providing business agility. In its unfinished state, a merely functional ERP system will not sustain a total enterprise information ecosystem that permits information to flow from its point of entry into the company's data stream to all of the places where it will be used without cumbersome intermediate steps. In a world of instantaneous customer-company and company-supplier interaction via the web, immediate access to correct, complete, and consistent information is the strategy for enhanced profitability and agility.

This vital information is stored within a company's database, and almost all of the large enterprise systems are built on relational databases. These databases are extraordinarily complex, often with thousands of tables and millions of columns that may be related to each other. There are four types of data: _seed data, configuration data, master data_, and _transaction data_. Because of all the relationships in an enterprise application, changes in the seed data or the configuration data impact all of the other data in the system, meaning that one small change in a configuration parameter could result in hundreds of thousands of changes in the system. When implementing an enterprise system, companies set up the parameters of the system. Initially, the system may allow changes to the setup, and users often go through several conference room pilots (tests) until the application is configured in a desired way. However, after the system is configured and the user has begun to enter transactions, most of the configuration data can't be altered because it is difficult to permeate the changes throughout the rest of the database. This leaves the business in a state where the applications support the current business, but not the subsequent

changes that all companies go through.

The ERP implementation process for multi-national corporations with hundreds of sites illustrates the frequency and magnitude of these changes. When I consulted, my team and I would implement at the first site and go into production successfully. We would do the same at the second site, but by the time we finished at the third site, the company had undergone a major business change like a merger or acquisition, had experienced a divestiture, had to comply with new statutory or regulatory requirements, or had gone to the Euro as a functional currency. We had to go back to the first site and reimplement in order to change the configuration parameters. The reimplementation often took as long and cost as much as the initial implementation, and often, by the time we were finished again, the business had changed again. Thus, while the enterprise implementation was considered "finished," most emphatically, it was not "done" because the company was not able to recognize the benefits of its enterprise application.

As described more fully below, there is a method for moving enterprise applications from "finished" to "done." The method operates at two levels: the ***individual application level*** and what might be called the ***enterprise ecosystem level***. Together, this method permits the company to maintain an interrelated series of mature applications that provide complete, consistent, and correct data that is efficiently delivered and maintained. If a change is required, it should only need to be made in a single place and from there would flow through the entire applications ecosystem. That maturity of each application and a similar maturing of the set of applications to efficiently support the company's business processes, in turn, permits adaptive modification that responds quickly and easily to any change – the desired level of agility.

Still, even a mature system has an ongoing life cycle necessitated by the simple fact that the business environment is never static, which is what prompted the need for agility in the first place. For example, adding a new product line or differently regulated market segment will require adjusting existing applications and may call for adding new applications, triggering a cyclical process. As new applications enter the lifecycle, the organizational ecosystem may take a temporary step back in its level of maturity until the new application reaches the "finished" stage, but it is not yet "done" enough to fluidly support ongoing business changes.

Method of Getting Enterprise Applications from *Finished* to *Done*

Document the Application Portfolio

An inventory must first be created. It should include each of the existing applications and identify the users, major functions or business processes that the application supports, data usage in the application (including create, read, update, and delete), the business unit(s) supported by the application, costs related to support and maintenance of the application, and expected life of the application.

Determine the Initial Configuration of Each New Application

As each application is implemented, the business rules dictate the basic parameters or configuration of that system. Those business rules might include supplier terms, customer credit limits, aging buckets, min and max reorder points for inventory, and the life of an asset. This is often the stage when companies say they are finished. The new application is ready to use to conduct day-to-day operations.

Stabilize Each Application

Global standards for the enterprise must be defined, and adjustments or changes made to all data within each existing application must conform to those standards. These standards might include a global chart of accounts, naming standards, pricing standards, catalog standards, compliance standards, and security standards. Standard business processes are all defined and each application is evaluated to determine its level of adherence. Each individual application becomes stable, and the applications taken together become consistent across the enterprise with all duplicates identified across all systems, combined and removed for all configuration data and master data.

Synthesize the Applications within the Enterprise

No enterprise application operates in a vacuum. During this stage, the applications of the enterprise work together with the information flowing seamlessly from one system to another, the data shared among applications, and redundant applications consolidated so that each business process is only supported by one application. During this stage, a master data management program is put into place so that there is a single source of data, and that data is created, updated, and deleted in one place. It is during the synthesis process that both the data and the processes become complete and correct and there is a single source of truth, a single way of doing business throughout the enterprise.

Connect the Applications to External Sources and Destinations

In order to be effective, both the enterprise's data and processes need to interact with events from outside the enterprise, whether it is accepting an order automatically from a customer, purchasing from a supplier, or complying with external statutory and reporting requirements. If there is a single source of truth for each data type within the enterprise, it is easy to communicate that information to the outside world. For example, if there is a standard product catalog, then customers can order from that catalog and be assured of product consistency whether they order from a storefront or order on-line. If inventory information is consolidated, then orders can go to suppliers on a planned demand schedule without the fear of over or under supply.

As these methods are deployed throughout the enterprise, the applications can be changed to reflect business changes. Because the applications portfolio has been rationalized and optimized, the way to effectuate most business changes (credit limits, new compliance requirements, altered currency operations, etc.) is easily identified and does not require substantial effort. The enterprise applications take on a new value because they now can be used to develop a strategy to undertake major changes like a merger or acquisition or a divestiture. The enterprise applications, instead of being the barrier to change that they were in a "finished, but not done" stage, literally contribute to instituting strategic changes because they provide a global view of the organization and the decision makers have access to complete, consistent, and correct information. If a company is thinking about acquiring another company, it is easy to determine whether to keep the applications because they add to data and processes that don't already exist in the company's portfolio. If a decision is made to sell part of the company, if all the information is in one place, the company is able to determine the impact of the change by modeling scenarios with and without that part of the company. As an enterprise is able to use its applications to enable business changes, those applications become a success and move from "***finished***" to "***done***," a mature state that allows them to respond to business change with agility.

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.