IMPLEMENTING AND INTEGRATING PACKAGED APPLICATIONS

- Traditional packaged applications are too often virtual prisons, locking away the information and business processes. The difficulty in accessing this information from within these popular applications is the primary reason for the increased interest in enterprise application integration (EAI). Correctly applied, EAI frees this information, making it available to other systems or end users who may need it.
- Although packaged application vendors have begun to see the value in opening up their architectures and technologies by allowing external applications and middleware to access their data and methods, they continue to do so using proprietary approaches. As a result, EAI architects and developers must understand the ins and outs of the packaged applications they are attempting to integrate. SAP, PeopleSoft, Baan, and Oracle represent only the tip of the iceberg. Hundreds of packaged applications are available, some easy to integrate, most difficult.

WHY PACKAGED APPLICATIONS?

- For the past several years, there has been a "refocusing" from custom applications to packaged applications. For many developers this represents a significant change in paradigm as well as process. What it ultimately represents is a tradeoff in the types of problems both application architects and developers are solving. A tradeoff, unfortunately, of technology and opportunity.
- Reasons for the rise in packaged applications:
  - First, organizations share many of the same business processes and services. Such processes as accounts receivable and accounts payable don't vary much from company to company, nor do purchase order systems or inventory control.
  - Second, it is finally possible to reuse common services through common applications. Historically, the software development industry has done a poor job of sharing objects, functions, and components.
  - Finally, there is a reduction in the time-to-delivery and the number of quality problems when dealing with packages.
- Packaged application helps organizations in need of a way to take the burden off their IT support teams while providing an improved end-user experience.

Benefits of Packaged Applications

1. Ensures a consistent, stable and reliable standard environment for the organization
2. Reducing support costs - When an application is packaged, it must have passed through multiple rounds of testing before it is rolled out to live users. If there is any issue with your application, it is very unlikely that it could slip through discovery, packaging and UAT altogether without being noticed.
3. Packaging once and distributing everywhere - A great benefit of application packaging is that once an application is packaged based on the requirements provided by the business, that package can be distributed everywhere.
4. Less business disruptions - package installation, uninstallation and upgrading can all be done silently in the background, without interrupting or prompting an action from the end-user. Most of the time, business users will not even notice that something got installed on their devices until they see the application shortcuts.
5. Easier application management - Once the application is packaged and set up in the configuration tool, then it's just a matter of adding the devices and/or users to the corresponding collection and managing it accordingly.
6. Effectively running application inventory and reporting - Suppose that an application worked without any issues, but a recent change in the environment broke the application or is preventing it from working as expected. Using a deployment tool like configuration tool to distribute packaged applications can identify the devices/users affected straight away and have a clear picture of how big the impact is.
7. Reducing security risks - Since the package installation, uninstallation, and upgrading are managed through the configuration tool, the vast majority of business users will not need

admin rights - which will lead to a low-security risk. As it is common knowledge, the least amount of people that have access to admin rights, the more secure an environment is.

8. Mitigating uncontrolled software installation - No admin rights for business users means they will not be able to manually install any application on their own. The application must either be installed using the configuration tool which is the preferred form, or by involving IT to manually install the application for them.
9. Customizing packages to suit business requirements - It is very rare for business users to have technical knowledge about the software they use. Generally, they just use application software as helpful tools to complete certain tasks in their job.

INSTALLING PACKAGED APPLICATIONS

- Preparing for a package installation and integration project demands a detailed understanding of the user requirements and business issues. It requires the answers to many questions. For example, how many end users will use the system on a daily basis? What are the hardware and software requirements for the package? What about sharing information and processes with other external systems? How long will it take for users to learn the new software? Each of these questions opens a minefield of other questions and problems. The difficulty and complexity of the task make it a good idea to document these requirements, feeding them back to the user to assure there is no miscommunication.
- A successful transition depends on a great deal of coordination. If this coordinated effort is not begun early in the installation and integration process, then the project is at risk before it has a chance to get off the ground.

- **Business drivers**
  - The lack of a clear vision regarding the business drivers is another potential project killer. Bottom line—the package must somehow solve some problem and thus make money for the company. If it doesn't, the company should reject the installation before it begins.
  - For example, if the sales department is not able to keep up with customer demand, then there is a genuine business problem. In this case, the value of the business driver is having a faster system in place to keep up with customer demand. As a result, the company will make more money. Using this as the "value proposition," it becomes straightforward to put the metrics in place to assure that the system lives up to expectations after going live.

- **Architecture Drive Success**
  - Neglecting the application architecture is a common problem during installation. Too often, the organization moves forward on the assumption that a packaged application installation effort does not require any formal application architecture. While it's true that installing and integrating a packaged application presents different architectural problems than a custom application, there are still technical problems to solve.
  - A common difficulty arises because most organizations don't accept the package as bundled. Instead, they make modifications to the business service and interface layers of the packaged application. Such modifications are often equivalent to a reengineering and application development activity.
  - Requirements must be gathered, interfaces designed, and business processes redefined. Also, a plan must be formulated to use the proprietary technology of these functions. Most vendors of packaged applications understand the requirements of application architects. As a result, they often bundle design and development tools.

- **Testing what has already been tested**
  - A rigorous testing process assures that the newly installed system is meeting performance, interface, and business expectations. This is especially important in an EAI problem domain where the fundamental issue is not so much that the system performs well by itself, but that it performs well when sharing information with any number of external systems.
  - There are three levels of package testing: performance, process, and interface.

    1. **Performance testing** is a direct examination of the throughput of the packaged application as well as the

entire system it is connected to (Figure 1). For example, how long will it take an order to appear on the warehouse PC after it has been entered in sales? How long does it take the package to locate a customer in the database? What about sharing information with external systems?

2. **Process testing** is a quick look at the accuracy of the system, as well as its ability to function at the level required by the business. For example, does 2+2=5? Or does the inventory report reflect the data feed into the system? Within an EAI problem domain, it is also necessary to validate that the information remains correct as it moves from system to system. This is most important when testing systems that have undergone significant customization and therefore have new processes that have not been tested by the package vendor.

3. **Interface testing** provides a quick look at how well the interface works with the human beings that have to use it. Too often, such testing is minimized or overlooked completely. Ultimately, if the end user has trouble with the interface, there will be problems with the system. Therefore, it is wise to work directly with the end users during the entire installation process, gathering requirements and assuring the user that the requirements are met in the final product.

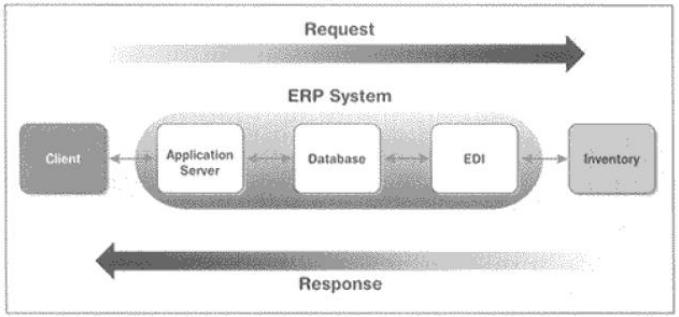

Figure 1. Performance Testing of an Inventory System

TESTING TOOLS FOR ERP

- Most packaged application vendors understand the need for testing. They provide either their own testing tools or interfaces to external tools. For example, SAP provides Automated Testing Tool interface (BC-ATT), a combination of SAP graphical user interface presentation server software and the Desktop Developer Kit interface. BC-ATT provides third-party testing tool vendors with the ability to execute under Windows 95 or Windows NT on SAP's Presentation Server PC, which communicates with the R/3 application server.

Implementing Specific Packages

SAP

- SAP uses a thin-client, three-tier architecture that is essentially proprietary. The real processing occurs at the middle tier, or the application server (Figure 2). R/3 leverages a transaction model, using a proprietary transaction processor. The R/3 clients are basically data collectors for the middle-tier transaction processor. The client defines its behavior by invoking transaction services on the middle tier. SAP R/3 uses off-the-shelf databases such as Oracle and Sybase for the database layer.

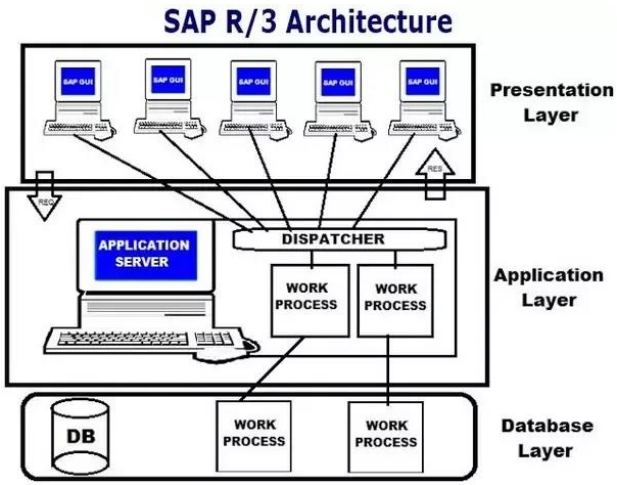

Figure 2. The Architecture of SAP R/3

- SAP's advance business application programming (ABAP) Development Workbench includes a repository, editor, and dictionary, as well as tools for tuning, testing, and debugging R/3 applications. This is where ABAP developers will customize the SAP software to meet the exact requirements of the enterprise. SAP layered this tool deep into the R/3 middleware layer. As a result, it is able to communicate with the application server as well as with the R/3 clients.
- SAP offers an implementation methodology called AcceleratedSAP. AcceleratedSAP is divided into several phases, including Project Preparation, Business Blueprint, Simulation, Validation, Final Preparation, and Go Live & Support.

People Soft

- PeopleSoft provides PeopleTools. Like SAP's ABAP, PeopleTools provides a set of methodologies and tools for installing and customizing its ERP application.
- PeopleTools is a rapid application development environment, which provides both application architects and developers with the ability to tailor PeopleSoft applications to the business requirements of the enterprise.
- Features of PeopleTools include a workflow designer, development subsystem, administration tools, reporting tools, and integration tools that link other products to PeopleSoft.

REFERENCES:

Fastrack IT Academy (2019). *SAP business one – Basic.* Fastrack IT Academy.

Lam, W. (2017).*Enterprise architecture and integration: Methods, implementation, and technologies.* Information Science Reference.

Linthicum, D. (2017). *Enterprise application integration.* Addison Wesley