A Practical Methodology for Integrating Functional Process Design and Data-Driven Packaged Software Deployment

John C. Goodpasture, Director, United States Operations Project Office, Lanier Worldwide, Inc.
Thomas N. Mangan, Vice President, Information Systems, Lanier Worldwide, Inc.

Introduction

As the millennium approaches, opportunities to change business systems are all about us. A project common among many information systems managers is deployment of "packaged software" to replace customized legacy business systems and to solve the "year 2000 problem." Increasingly, a large number of these deployments are accompanied by redesign of business practices to take advantage of the functionality of these package solutions. However, such "joint" projects have not been all that common in the past. What has changed?

The authors suggest that a confluence of three elements make changes like these possible:

- the rise of practical software package solutions to replace what has been custom, or at least customized, mainframe data management solutions
- the compelling need to modernize for the year 2000
- the widespread employment of functional process design techniques to achieve improved and more competitive business operations.

The coming together of these three elements is a unique opportunity for information systems managers to exploit both the "business integration" possibilities enabled by the technology and the urgent need of year 2000. However, packaged software and process design methodologies were developed independently. They have been introduced into the business community on a practical scale only in the last ten years or so. Integrating them is more art than science and is largely at the behest of the business project team. This paper provides a practical methodology for integrating software package deployment with business process redesign in one Work Breakdown Structure (WBS). This methodology has been employed by Lanier Worldwide, Inc., in its United States (U.S.) Operations enterprise, a $1000M business unit providing sales, distribution, and service of office products and services.

The Nature of Packaged Software and the Year 2000

The thesis of this paper is that the concurrency of the year 2000, packaged software, primarily client server software, and mature business engineering techniques is a unique opportunity for information systems managers and process managers to work in an integrated fashion. Exhibit 1 summarizes some of the important points of the integration of business engineering, client-server packaged software, and the "mainframe" legacy environment.

Prerequisites for Project Methodology

There are two prerequisites, which make project implementation and deployment go more smoothly. The first is flowdown, or decomposition, of the business goals to the project level. The authors think of this as business architecture. Exhibit 2 shows the business architecture flowdown; the three boxes that are in bold outline are those that are typically the focus of the project team. The second prerequisite is completed designs of the multifunctional strategic processes that support the business. An example of such a strategic process is "Sales Order to Cash." This process touches several functions from the sales force to the back office. The metrics of such a strategic process transcend many departmental objectives. There are many established ways to go about synthesizing a future state process and retaining the desirable features of the current state, while improving the likelihood of changes necessary to reduce production constraints, improve cycle time, reduce process cost, and increase throughput.

Project Methodology

The methodology described in this paper is purposely designed to accomplish several goals.

- Exploit the opportunity of Year 2000-driven technology improvements to improve the business and, by extension, the processes that drive the business, being responsive to issues raised in Exhibit 1.
- Provide a harmonious functional and technical solution that will be widely embraced when deployed, thereby enhancing the possibility of successful organizational change.
- Provide seamless functional integration between legacy information systems and new packaged software.

Proceedings of the 29th Annual Project Management Institute 1998 Seminars & Symposium
Long Beach, California, USA: Papers Presented October 9 to 15, 1998

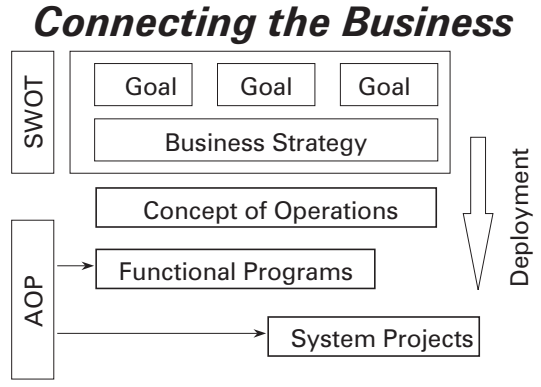
Exhibit 1. Integrating Process Design with Packaged Software and Legacy Processors

Business Process Topic	Packaged Software Implications	Legacy Implications	Year 2000 Implications
Process Design Question: Modify the business or modify the package?	Modifications to delivered Package may drive "cost of ownership" to impractical levels	Without modifications to the business, it may be necessary to maintain either legacy on-line, batch, or both, requiring potentially complex interfaces to the Package	If legacy is retained, remediation of Y2K problems is required. Creates dependency with package implementation
Training & Transition to new process	Package upgrades & maintenance will necessitate training and transition during the package life cycle	Retest and recertification of legacy interfaces is needed at each transition	Year 2000 should be made transparent to the User
User Job Design to achieve the labor benefit of the new process	Scripting the Package without time to consider job design alternatives may lead to "Legacy with a Mouse"	Redesigned job's "fit" to the legacy may no longer be "optimum", i.e., long standing legacy customizations may be impacted	
New Process Cycle Time: Does the business run faster?	On-Line transaction speed, allocation of processing to batch processes, length and frequency of batch processing job stream	Legacy on-line and package on-line may have to co-exist, perhaps transparently to the user. Job streams for Legacy batch cycles & Package batch cycles require coordination and integration of scheduling which may be difficult, and impose constraints on the package which limits process effectiveness	

Obviously, a cross-functional design, development, and implementation project team is needed. The size of the team and the skill mix are project specific, but the author's experience is that the following elements need representation:

- information systems software developers, data and database administrators, job stream experts, information systems administration experts, network infrastructure experts, and client server infrastructure experts
- functional experts in the domain of the processes, typically spanning several functional departments, including those who can assess policy and procedure impacts
- functional and technical expertise in report writing (Here, it is assumed that on periodic cycles, the batch processes of either the legacy or the client-server systems will produce various types of reports for system administrators, business process control, and operational management.)
- functional and technical expertise in training, system deployment, and change management.

Exhibit 2. Business Architecture



Methodology Defined

The methodology diagrammed in Exhibit 3 integrates typical elements from a business process design project with typical elements from an information systems project as shown in the WBS in Exhibit 3.

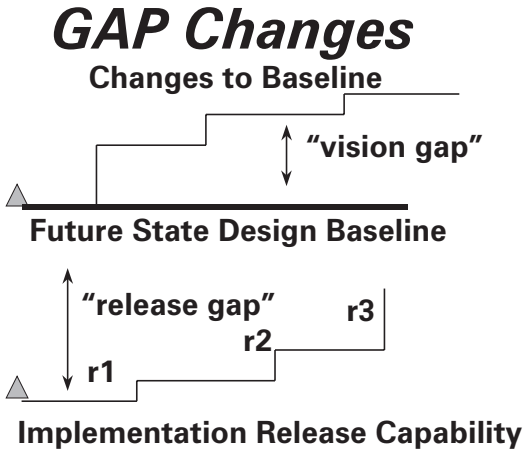
Scope is the objective or problem on which the new system change is trying to focus. This establishes the boundaries of the problem being studied. To be clear, the scope should define "what is" included in the new system, as well as "what is not" included in the new system. Once scope is defined, *Requirements* are functional statements, arranged hierarchically, and indexed by major topics. They are design independent, and singular [that is, not compound], and are not statements of benefits, policy, or procedure.

The *Concept of Operations* expresses the functions of the user community in the context of the system environment. Addressed in the Concept of Operations are the future

Exhibit 3. Project Methodology Steps

WBS	Work Package Leader	Deliverable
1.0 Scope & Requirements	Functional	Synthesis and evaluation of a requirements set, and concept of operations for the future state. Comparison of the "future state" with the capability of the Package yields the Gap Analysis. These deliverables jointly serve the needs of both the functional process design and the data system design.
2.0 Process Mapping	Functional	Identification of dependencies among the cross-functional requirements, translation of requirements into scripted, functional processes, and "personalization" of the package software with configuration data to conform to process requirements
3.0 Development	Technical	Application Development, Mapping and Conversion of legacy data into the package application which enables execution of scripted functions to validate business scenarios
4.0 Validation	Technical	Functional validation of the system applied to the business processes, and technical validation of the system to the Technical Performance Measures by Production Modeling
5.0 Training & Support	Functional	Preparation of, and deployment to the user community and support communities, the training and transition deliverables for the new processes and the new system technology

Exhibit 4. Gap Analysis



state cycle time, batch and online traits, reports, production throughput, and data management.

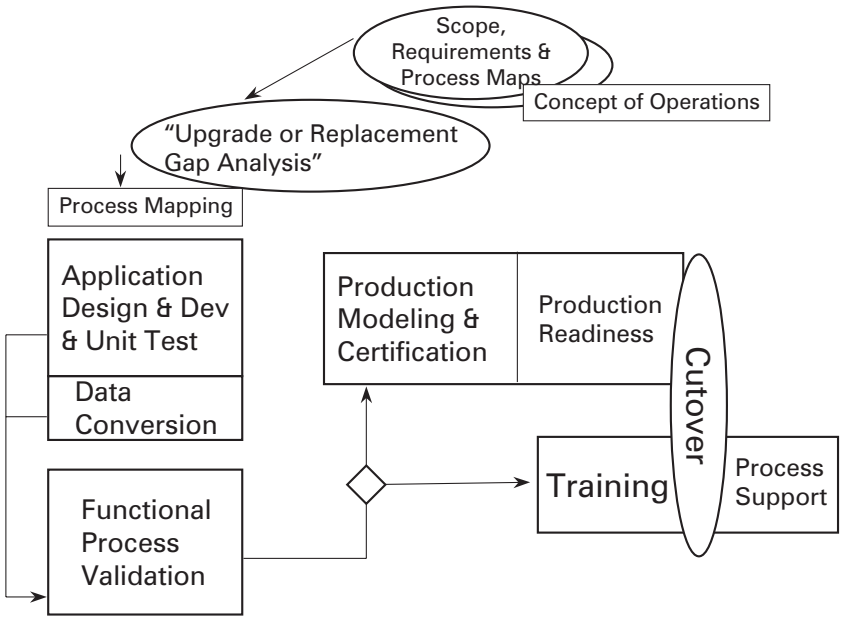
The *Gap Analysis* expresses the functional differences between the requirements and the delivered capability of the package. When completed, the gaps are graded in terms of importance to the value of the project, the risk and cost of implementation, and the effect on schedule. With the gap results, the development of a "dot release" strategy begins. Although generally thought of as an information systems function, in fact the "dot release" is as much a process owner concern as an information management concern. Exhibit 4 illustrates the concept of the "vision gap" and the "release gap."

Process Mapping delivers flow charts, derived from the strategic business processes, which are used to allocate specific capabilities of the online and batch capabilities of the package software and the legacy to operational tasks. All business reports are explicitly accounted for in the process maps. However, these maps are not a system schematic. Therefore, there is not a need to segment the maps to conform to hardware boundaries. Nor is the process map an organization chart. Thus, there need not be a conformance to the administrative boundaries of the organization. However, the maps do show the way that the work gets done. The mark of a good map is that future state jobs can be overlaid without ambiguity.

Application Design, Development, and Unit test is the traditional engineering activity. But, the unique aspect is the development of interfaces to the legacy environment and conversions of the legacy data. Therefore, as shown, data maps must be in place and desired configuration of the application must be known before proceeding with application design (configuration is a term used to denote defining, or configuring, the data definitions of all fields and tables in the application). Decisions are made regarding what legacy data is needed in the future state, what is to be "abandoned," and where data is going to be sourced (versus derived) in the enterprise.

Functional Process Validation is essentially a functional string test of system modules, components of the concept of operations, and of specific enumerated requirements. This is a unique step, which integrates business process engineering with traditional information system methods. More detail on this step is provided in the following section.

Exhibit 5. Project Methodology



Production Modeling and Certification is a multifunctional activity to validate the manner in which the system will perform in production. This also is a unique step, representing the final integrated test of functional and technical requirements.

The remaining steps are straightforward. They are given in Exhibit 6.

Integrating Functional Process Design and Data-Driven Packaged Software Deployment

Of the several steps shown in Exhibit 6, four are key to integrating process design and packaged software development.

Gap Analysis

The prerequisite for Gap Analysis is requirements synthesis. In the author's experience, the most practical approach to Gap Analysis is to develop a matrix to hold a coding scheme of requirements versus package functionality. The essential idea is to work from the requirements to the package, as there may be many functions within the package that are not requirements and need not be invoked, tested, or made operational. The Gap Analysis task often exposes the so-called "Boundary Gap" as illustrated in Exhibit 7. Two issues are shown: (a) The package capability is greater than the business requirement, requiring "procedure control" to insure business boundaries are not exceeded; (b) The package capability falls short of business need, leading to a tradeoff to "modify the business or modify the package." To modify the business usually involves executive intervention in order to change policy or make substantive changes to the "business rules." In the author's experience, substantive modification to the package is almost universally a wrong decision. This is because it is impractical in an enterprise of any size to maintain highly modified packages at the pace that new releases come from package vendors. Exhibit 8 illustrates an important consideration in making the decision to modify, or not, the Package. Those elements of the enterprise data system, which "touch the customer" or make the business unique in the customer's eyes, are unlikely to be good candidates for a package solution. Thus, the most practical use for packaged software is where "vanilla" installations are likely.

Configuration, Mapping, and Data Conversion

This step presumes that there is legacy data that will need to be represented in the package software.

There are at least two possibilities. One is that the source of a data element is to remain in the legacy, but a derived version must be in the package. The second is that the legacy data must be moved from the legacy to the package where it will then become the source. Data maintenance is always done at its source. The authors recommend the following approach.

- First, determine the configuration data needs of the package. This is a functional task. Develop the configuration

Exhibit 6. Remaining Methodology Steps

Step	Definition
Production Readiness	Technical task to rehearse and time the job stream, and all pre-cutover tasks
Cutover	Execute the job stream before turning on on-line production of new system
Training and Process Support	Train the trainer, user training, change-management communications, follow-up Help Desk, and SWAT teams to assist in production operations

Exhibit 7. Boundary Gap

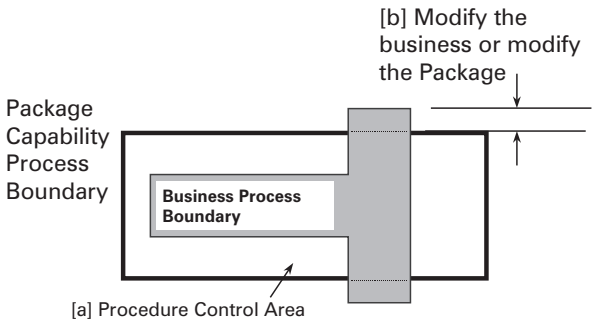


Exhibit 8. Custom vs Package ERP

data dictionary, if not delivered, of the data elements for all panels in the package that will be implemented.

- Then, working from the package toward the legacy, identify legacy data that will become the data elements in the package solution.
- Map legacy data to the package, write conversion interfaces if necessary, and then convert the data. The conversion methodology requires some technical trade-offs. That is, any data to be converted by means of an interface to the legacy data requires that the interface be written and tested and that the conversion be validated. If the volume is small enough that a manual reentry of data into the package is practical, this is often more cost effective than designing and implementing a conversion interface. Regardless of the conversion methodology, a key step is that while this technical work is performed, the functional team members validate that the legacy data not mapped is obsolete and can be abandoned. They also validate that incorrect, obsolete, and inconsistent data that occupies fields to be converted are corrected.

In most situations, the authors believe that Configuration, Mapping, and Data Conversion is the highest risk step in the methodology, leads to the most test time downstream, and is most likely to disrupt the cut-over to operations.

Functional Validation

Functional validation is for the most part a functional unit test as defined in the Project Methodology section. The steps in Functional Validation are as follows.

- The functional team writes procedural scripts used to exercise the Process Maps step by step.
- The technical team converts business data from the legacy into a validation database. This database is loaded with the configuration data defined by the functional team.
- Certain of the converted data are selected as datasets to exercise the scripts; other data needed for the scripts may have to be synthesized.
- Using the scripts, the functional team exercises segments of the process maps. In this step, there is not emphasis on timing or the continuity of scenarios. These attributes will be tested in the Process Modeling and Certification.

Process Modeling and Certification

Process Modeling and Certification (PM&C) is an integrated system test. It builds upon the Functional Validation defined above. However, in this step, online scripts, practiced in Functional Validation, are strung together

Exhibit 9. Lessons Learned

Item	Lesson Learned
Testing and Data "scope"	If the test data set is not sufficiently robust, many valid paths through the Package, and thereby through the Process, may not be invoked, leading to surprising performance at operational cutover. And, without sufficient volumes of data, or data "scope", cycle times may not be known
Data Integrity	Data variability [or "scope"] and "dirty data" make data conversion the highest risk task; many rehearsals of conversion and of batch cycles will be needed before formal cutover
Project Scope Creep	The greatest hazard of scope creep is to allow modifications to the package; a Modification Review Board is a good tool for controlling the demands for modifications
Scripting	Scripting of Process Maps to show validation should be held to a level of detail appropriate to the test; overly detailed scripts extend test time and may obscure achieving test objective
Training & Deployment	Training investment pays off. A good training program yields much less lost productivity when users begin production activity
Performance Testing	Developing a rigorous and disciplined test plan results in fewer performance surprises after initial implementation
Sizing the System	Over engineer most components of the technical infrastructure and create a reserve of disk and extra processor capability for unanticipated performance problems
User Groups	Establish a network group of peers which has done it before and solved the same problem
Release Planning	Don't implement the "dot 0" version of any major release. Test all minor "dot releases" for not only the proposed changes but also the features which worked before.
Transaction Integrity	Maintain the integrity of the transaction process. Don't break a natural transaction across legacy and packaged software boundaries, if possible. This creates complex two-way interface issues. Try to implement all the packaged software modules across a natural transaction flow, e.g., order fulfillment.
Scrutinize Consultants	With the shortage of qualified I.T. professionals, don't become the next training ground for a consultant practice. Check references of the individuals who will be on your project.

into scenarios and combined with batch processes. Thus, a production-like job stream is exercised, and interfaces to the legacy are tested. Timing, or responsiveness, is measured in this step. In fact, *Test and Tune* is a critical step in PM&C. Most of the failed attempts at packaged software implementations (or at least those that resulted in significant cost overruns) have been a result of lack of infrastructure planning and tuning, both network and client-server. Most legacy environments must undergo a complete redesign of their information technology infrastructure to achieve the performance metrics identified in the systems requirements. Implementation of a "standard client" is essential. Without a well-designed and disciplined performance testing and tuning strategy, the initial attempts to bring the new client-server system live will fail or, at a minimum, negatively impact the perception of the new packaged software in the eyes of the end user.

A practical approach to PM&C is to have a test instance in the legacy. Converted data is captured as a static test dataset. This permits repeatability of tests without data variations; daily, weekly, monthly, or other periodic aspects of the Process Maps are proven in various scenarios. To achieve all of the test goals, production volumes of data are used, date-sensitive data is aged, and synthesis of missing data might be necessary.

Programs to Support the Operating Environment

Referring back to Exhibit 1, operating "programs" are necessary to support the systems and processes, once operational. Although the range of programs can be quite wide, two are important to data-driven projects. The first is a program of Data Management and Data Maintenance, including application configuration data, which is essential to the integrity of a data processing system. Formal procedures in data management, appointment of data owners and custodians, and training in data management are requirements for good order in enterprise data. The second is a program of Process Map maintenance. This includes measurement and reporting of benefits, efforts to "internalize" the new process, and continuous improvement actions.

Summary and Lessons Learned

Lessons learned are about what was not known at the outset of the project, which can be usefully applied on the next employment of the methodology. Exhibit 9 illustrates some of the author's experience with this methodology.