Operational ERP Architecture for aligning Rollout Projects with Operations in global Template Programs

consulting4bit

June 7, 2013

Introduction

Global ERP template programs are characterized by a multitude of rollout projects that must be managed in parallel to operations. The operational ERP architecture is a key success factor for managing design, development and testing of all ERP system configurations effectively . If set up appropriately projects run smoothly in parallel to operations and typical global template program objectives like business process standardization and harmonization are well achievable. If not aligned with the overall program approach the risks are high that the operational ERP architecture will become a bottleneck for your template initiative. A bad design can even lead to a standstill of projects if inconsistencies in the ERP system landscape need to be fixed.

This document explains the aspects you need to consider when designing an appropriate operational ERP architecture. The operational ERP architecture focusses on the aspects that are important for running operations and projects in parallel. It must be distinguished from the strategic ERP architecture which focusses on strategic business and IT requirements. This architecture is not in focus of this document. It will be outlined in general but apart from that it will be regarded here as a given. Different patterns of the operational ERP architecture will be explained in more detail. Decision criteria are presented to evaluate in which situations a specific pattern is recommended or not appropriate. The document concludes with a few general recommendations from practical experiences that are applicable to all patterns.

Options for designing the strategic ERP Architecture

Over the last 10 - 15 years years many companies decided to consolidate their ERP system landscape and started global template programs to deploy a new ERP system. A key decision in the beginning of such programs is to find the right strategic ERP architecture. The strategic ERP architecture refers to the long-term system landscape structure. It must fulfill business and IT requirements which are derived from the overall strategy. Any global ERP template initiative should start with a detailed study how to design this architecture. The design options found in practice are limited and focus on the dimensions for structuring the ERP system landscape. The following dimensions will be discussed here in short:

- Set up regional ERP systems
- Set up functional ERP systems
- Set up divisional ERP systems
- Set up one ERP system for all businesses and functions
- Implement a mix of different ERP types

The approach to set up regional ERP systems was very common in the last years. It was driven mainly by concerns that one global ERP system would be

too complex to manage and would impose high risks for the business in case of failure. Also the amount of transactional data of some companies is so immense that is was considered as too much for one system. Business concerns referred to aspects like flexibility which was seen at risk with one large IT solution that might not keep pace with different business needs and different market development speeds. Some of these concerns are today no longer vaild as the scalability of ERP matured. Many IT organizations are today capable of running even very large solutions reliably. The flexibility discussion is still ongoing. Regional systems do not solve this issue as regions itself are very heterogeneous.

Functional ERP systems have often been chosen by companies who had already organized their business entities according to functions, i.e. have pure sales or production affiliates. The biggest challenge with functional ERP systems is to manage the integration between the business entities which has a large impact on the design of financial and supply chain processes.

The divisional approach was favoured by companies with a very heterogeneous business. Typical for those businesses are M&A activities which happen frequently. Having enough flexibility within the divisions was therefore rated very high. But as divisions can be managed like independent companies the approach of building divisional ERP systems is not fundamentally different from the scenario "One global ERP".

In recent years a new approach came up which focussed on a combination of different ERP solutions for large and small affiliates of a company. This approach is driven by costs of ERP solutions. Often solutions like SAP ERP are too expensive for small affiliates with just a limited set of requirements. Cloud-based ERP solutions are becoming more and more attractive for these affiliates. The biggest challenge of using different ERP solutions is their integration, similar to the functional approach.

It is out of scope here to describe the decision process how to find an appropriate solution for the strategic ERP architecture. It must finally support your business and IT strategy. And it must be implemented in a way that it supports the alignment of rollout project activities with operations. How such an alignment can look like will be explained in the next sections. The scenario "One global ERP solution" will be used to explain the alignment process. The biggest influencing factor for the alignment process is the software architecture of the ERP system. As SAP ERP is the market leader in ERP systems it will be used in the following to explain the alignment process.

Objectives for aligning Rollouts with Operations

After the strategic ERP architecture has been finalized the foundations for the global template program are laid. During the template build phase the operational architecture is prepared to ensure that after the template is deployed the first rollout projects can start. The critical phase for the operational architecture begins after the first Go lives of the pilot projects when the next wave

of rollouts starts. Now projects and operations must be managed in parallel. The operational architecture must ensure that

- operations are not jeopardized by projects,
- it is possible to run several projects in parallel to operations,
- configurations (include customizing + developments) do not lead to ERP system inconsistencies and
- projects and operations can work largely independently from each other with a minimum amount of reconciliations.

The operational ERP architecture is your infrastructure for managing design, development and testing of all ERP system configurations effectively. The patterns for this architecture you will find in practice are explained in the following section.

Patterns for the operational ERP Architecture

The simplest ERP system landscape includes three systems: A development system DEV, a quality assurance system QA and the productive system PROD:

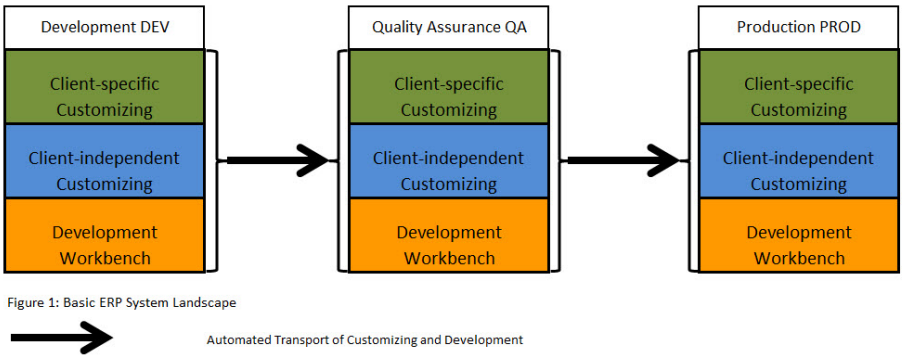

Figure 1: Basic ERP System Landscape

For understanding the challenges of running projects in parallel to operations it is important to understand the software architecture of ERP. Here we use SAP ERP as example. The software architecture of SAP ERP distinguishes three sections:

- The development workbench for creating programs
- The client-independent customizing for configuring system settings for all clients
- The client-dependent customizing fot configuring system settings for a specific client only

The client is a technical element that divides an ERP system into separate sections. A SAP ERP system can have several different clients. If customizing settings are client-dependent this means that they include the client number (3-digit key) as part of their key. If the ERP system has more than one client the

client-independent customizing and the development workbench can be used to develop and customize settings that are relevant for all clients. By this software architecture common and individual configurations shall be enabled.

Using this basic operational ERP architecture for global template programs is usually not recommended. A main reason that speaks against this architecture is that all projects and operations use the same client. This requires high reconciliation efforts between projects and operations already when projects start and has technical risks. A frequent problem in practice is that the same configurations are changed by various projects. If these changes are not synchronized this can lead to inconsistencies after transport. Project configurations undergo a series of changes in the design phase and it is not reasonable to mix these with operational changes which are usually more clear-defined.

A better approach for global template programs is to use different clients for projects and operations. An exemplary ERP architecture with two projects and operations could then look as follows:

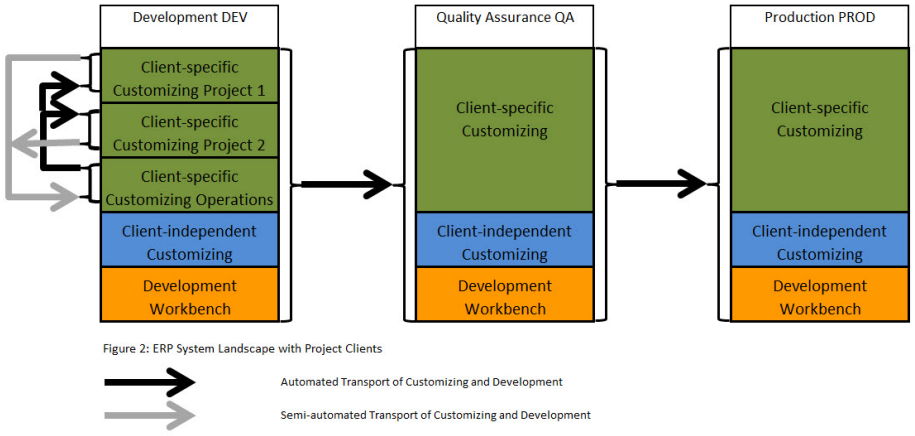

Figure 2: ERP System Landscape with Project Clients

This ERP architecture uses different clients for projects 1 and 2 and for operations in the development DEV system. This allows that the projects and operations can configure business processes by using client-specific customization independent from each other. Only client-independent customizing and developments need to be reconciliated between all parties. From experience, a company can implement 80 percent or more of its required functionalities by customization only. Some exceptions apply for certain industries or countries that are heavily regulated. But less than 20 percent of functionalities have to be made in the sections that are client-independent. Risks due to interferences between projects and operations can thus be reduced considerably

In order to be effective a few procedures need to be followed in an operational ERP architecture with several clients. As the length of the design phase of a rollout project varies and can easily cover 3 to 6 month it must be ensured that changes from operations are regularly transported to the project clients. Main reason for this is that project tests should be done in a system environment which resembles the actual operational settings. Tests in the design phase in the development system are functional tests which are frequently repeated and

usually do not need a formal documentation. Changes from operations which are moved to the project clients are therefore less critical for the project schedule. Settings from other implementation projects are not moved between the clients. They remain in their client until the design phase ends.

At the end of the design phase a consolidation phase starts. All customizing settings of the projects are consolidated and moved to the operational client (sometimes also called "Golden Client"). This consolidation should be done by a central team. As the consolidation process can require a resolution of implementation conflicts (e.g. duplicate keys in customizing tables) it is not fully automated and includes transports as well as manual activities.

Both operational architectures discussed so far have client-independent customizing and the development workbench in common. Some companies favour an approach for building up a template that starts with a lean template and adds large chunks of functionalities in the first rollouts. These chunks also include developments. For those companies an approach to set up a separate project development system can be reasonable as it further minimizes the risks of conflicts between projects and operations. In a project development landscape only rollout project configurations are done. Configurations from operations are moved to the project system via transports. Configurations from the projects which also include developments are moved to the operations client after they are finalized and tested.

There are two variants you will often find in a project system landscape. The first variant includes only a project development system while the second variant includes an additional quality assurance system. The main motivation for having a quality assurance system is that the projects can do more and better tests and will deliver a better configuration quality at the end of the design phase. The downside of this additional system is that the architecture is more complex and projects are synchronized later with operations. Both variants for architectures with separated project development and quality assurance systems are illustrated on the next two pages:

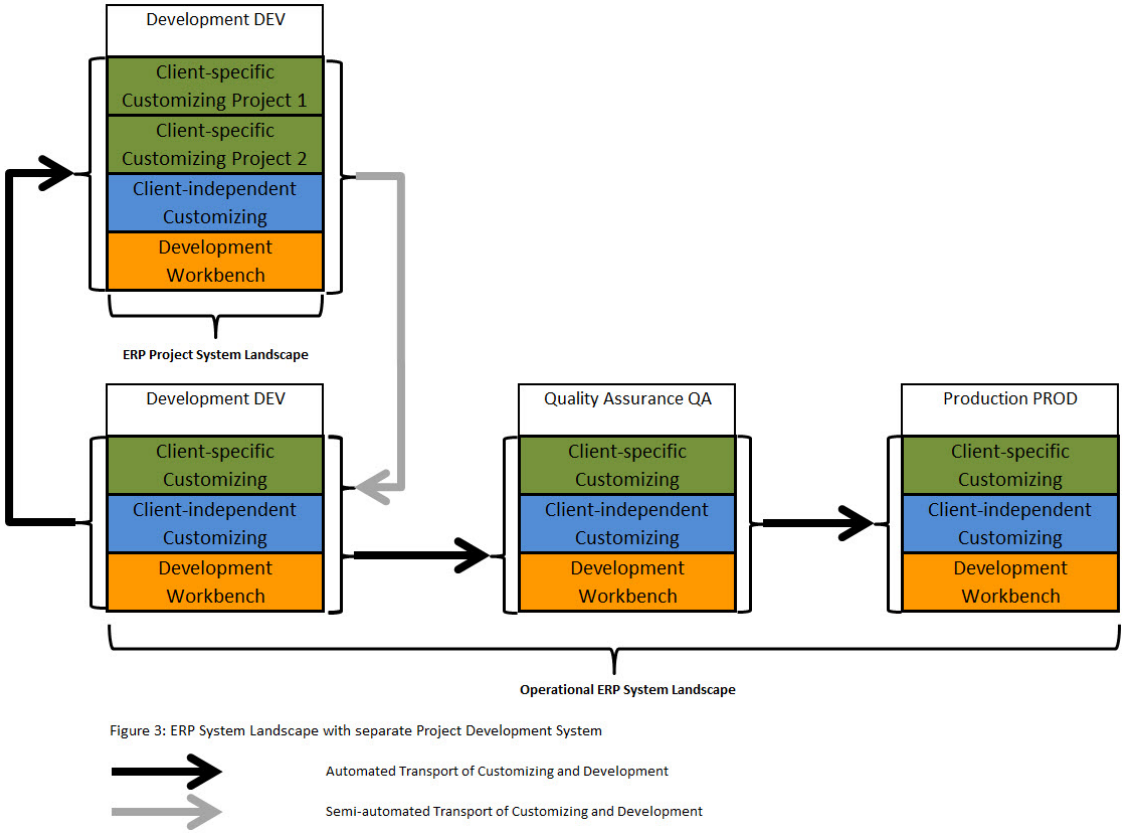

Figure 3: ERP System Landscape with separate Project Development System

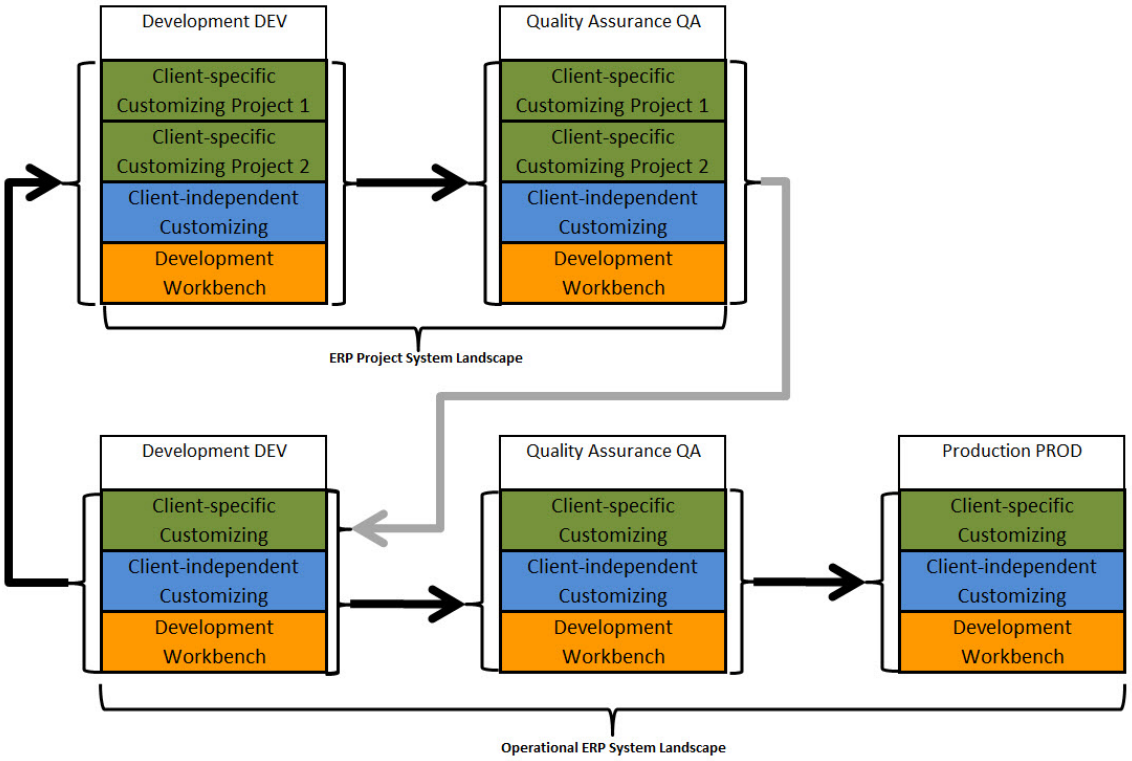

Figure 4: ERP System Landscape with separate Project Development + Quality Assurance Systems

Automated Transport of Customizing and Development

Semi-automated Transport of Customizing and Development

The next section will list up and discuss decision criteria in which situation which operational ERP architecture will be appropriate for your program. Except the first model which has obvious deficits the fitness of the other three models depends on the situation.

Assessment Criteria for selecting the operational ERP Architecture

Four different operational ERP architectures have been discussed so far. The following list of criteria can help you to find the best approach for your template program:

- C1: Completeness of Template Content
- C2: Expected Amount of local Functionalities in a Rollout
- C3: Availability and Experience of global Implementation Team
- C4: Expected Quality of Project Implementations and Tests
- C5: Number and Complexity of Projects that run in parallel
- C6: Status of Change and Release Management
- C7: Degree of Standardization

C1: The completeness of template content describes to what extent the template functionalities cover what is needed during the rollouts. Companies who choose to set up a template very fast complement the content during the rollouts. In this case the template does only contains core functionalities which cover far less than the estimated 80 % of functionalities where you expect that they can be used globally. Large parts of functionalities are then added in the first rollouts. In this situation a project system is more appropriate than a N client ERP system.

C2: The expected amount of local functionalities in a rollout refers to the question how many new local functionalities do you expect in a rollout project. Even if the template content already covers 80 % or more of what is needed a rollout project could require specific local functionalities that are complex and large. Typical examples are rollouts to a country like Brasil which has very extensive legal requirements or the introduction of a complete new ERP module for a site. For such rollout projects a seperate project system is advantageous.

C3: The role of the global implementation team in a template project is to coordinate all activities that refer to global developments and configurations. The global implementation team consolidates what the different projects deliver and it has a quality assurance role. It is the guardian of the global development system and the golden client. Irrespective of which operational ERP architecture you use you definitely need such a team. But if you do not yet have this

team then starting your template project without a project system is very risky.

C4: The quality you can expect from the rollout projects depends on the qualifications of the people responsible for configurations and tests. If you have an experienced team which has already done some projects a seperate project development or project quality assurance system might not be needed. If the project delivers a poor quality it is reasonable to use a project development and even a quality assurance system to limit impacts of bad configurations.

C5: Each template program needs a rollout plan and a strategy how to build the rollout waves. Project complexity, duration, locations or technical priorities are just a few examples for criteria that can be used for building the rollout plan. If your projects are small and less complex a separate project system might not be needed. But if you run a large program with several projects in different regions a project system helps to reduce risks and offers more flexibility for the implementation schedule.

C6: Processes for handling changes and managing releases are mandatory for global template programs. Often these processes are introduced when a global template is started. In the beginning of projects during the design phase many changes can be expected. A seperate project system landscape could be advantageous as a lean procedure for handling changes could be applied here.

C7: The degree of standardization measures how many organizations use the same process. Templates usually strive for a high degree of standardization. If a process which is used by many organizations is changed the impact can be large. If risks should be reduced to a minimum a seperate project system is the best approach. But also an N client ERP architecture can handle this. Apart from the technical aspects changes of global standards must be discussed between all stakeholders.

The following table summarizes the ratings for the different criteria:

Evaluation of Operational ERP Architecture Patterns

Evaluation Criteria List	1 Client ERP	N Client ERP	Project Dev	Project DEV+QA
Large and complex Rollout Program, Program is in Startup Lifecycle	↓	→	↑	↑
Rollout Program is mature, Rollouts have become Routine	→	↑	→	→
Experienced global Development Team available	→	↑	↑	→
Scope of Global Template Content is low, quick Growth by next Rollouts expected	↓	→	↑	↑
Scope of Global Template Content is high, no major new Content expected	→	↑	↑	→
Rollout Project Teams are unexperienced	↓	→	↑	↑
Rollout Project Teams are professional	→	↑	→	→
Change & Release Management Process well established	→	↑	↑	→
High Degree of Business Process Standardization	→	↑	↑	→

The evaluation shows that the first model 1 Client ERP is usually not recommended. Even after a template program is finished you might have projects that reach a certain complexity where a separate client or even a project development system can make sense. Unless for very simple projects this model should only be used for operational changes running according to change request procedures.
The other three models have advantages and disadvantages. Their rating mainly differs according to the lifecycle your template program is in and according to the amount of experience your team has acquired. In a template startup phase with a team that does not have much experience with rollout projects a seperate project system landscape has clear advantages. But after two or three waves of rollouts you hopefully have a template that covers 80 % or more of what you need. Then a switch from a project system landscape to a N client system landscape could be made. Although it is possible to continue using a seperate project system or even a project quality assurance system you should always check if this landscape is still needed as it is more complex and expensive.

Summary and Recommendations

The discussed models for the operational ERP architecture represent common patterns. You will find further variations of these patterns. Which pattern you chose to build your operational ERP architecture will finally depend on

- the lifecycle your template program is in,
- the experience your organization has with managing implementation risks and architectural complexity and
- the degree of standardization you want to achieve.

Depending on the lifecycle of your template program you should usually start with a project system landscape and consider a simpler multi-client model after the first rollouts are done. A global team that reviews what the projects do and implements that in the golden client is a key success factor. It should not be bypassed by allowing local organizations to do their own develpments. This is not only very risky but also undermines the objectives of having globally standardized processes.

It is important that the concept for your operational ERP architecture is clear before the rollouts start. You should plan a dry run with your project team and discuss how a rollout project with its different phases runs through the operational ERP architecture. The project plan must reflect this as you will have specific activities in the different systems. As testing is very much interrelated with this architecture the test concept must be closely aligned with your concept. You can review the architecture after each rollout wave. But ensure stability while rollouts are in progress.