The Software Acquisition Life Cycle

Introduction

The previous module showed that IT applications are central to the quest by many organizations to fulfill their vision and mission. Moreover, most IT applications revolve around software and organizations are increasingly buying software, rather than developing it themselves.

The acquisition of any large artifact, whether it is an airplane by an airliner, a medical device by a hospital or an enterprise system by a manufacturing organization, is a complex process. Since complex processes are hard to manage, there is every chance of not completing the task correctly and ending up with a product that is not appropriate and, as a result, cannot be used. It therefore becomes important to find ways to handle this complexity.

There are a number of measures that an organization can put in place to manage complex projects. One obvious option is to hire better and more experienced personnel, and/or to use better tools to manage the process. However, probably the best way to handle the inevitable complexity of any large software acquisition project is to systematize the process. A more systematized process is likely to be more repeatable. Moreover, if every project uses its own methodology and tools, it becomes hard to see how the lessons that were learned in one project can be applied to another. Different projects may for example use different names for similar activities, or may even divide the overall task in different ways. By agreeing on the steps that need to be undertaken in any acquisition project, their names and goals, and the order in which they are performed, an organization can often significantly improve its success in such projects. Software engineers refer to systematic representations of the steps that have to be performed in complex software engineering projects as life cycle models.

In the remainder of this module, we will first discuss software engineering life cycle models, before turning to a software acquisition life cycle model.

Software Engineering Life Cycle Models

As said, software engineers refer to a systematized representation of the activities involved in the constructing of a piece of software as a life cycle model. Software engineers do not always agree about the best order in which to perform the various activities necessary to produce a piece of software that, when installed in its intended environment, will perform its intended function. They also disagree about whether these activities should be performed sequentially or whether there should be some iteration between the various activities. However, there is a broad agreement among software engineers about which activities have to be performed in any software
development project. Roughly, the activities involved in any software development project are:

1. Requirements analysis and definition
   Developing a specification of what the new software system is supposed to do for users (the services it is expected to provide), and constraints on the system’s operation and/or development (e.g., the hardware that it needs to run on, and any schedule and cost constraints). This activity is probably the most important in any software development process. After all, if one does not know what the software is supposed to do, then it will be nearly impossible to develop a system that is acceptable to the user. We discussed details about this in module 1 on requirements engineering.

2. System and software design
   Designing, in an abstract way, the system that the software is to be part of, and the software itself to realize the specification that resulted from the requirements analysis and definition

3. Implementation and unit testing
   The writing of the actual code that implements the design and the testing of the various modules that make up the new software

4. Integration and system testing
   Integrating the new software with the other systems that it needs to interact with, and testing of the new full system, typically on test cases that were specified during the requirements analysis and definition activity

5. Installation and acceptance testing
   Installing the software and testing whether the new system indeed meets the needs of the users

6. Operation and maintenance
   Running the system and maintaining it by fixing bugs and/or adding functionality required by users

There are many different software life cycle models but the main differences between the different models concern the order in which these activities are performed and whether they can overlap. For example, in the most traditional life cycle model, the so-called waterfall model, each activity has to take place in a distinct phase and each phase must be completed and signed off on before the project can move into the next phase. In iterative models, on the other hand, the software development team typically performs some requirements analysis, uses this partial specification to implement and
test a part of the system, delivers this to the user, and uses this partial system to develop a better understanding of the remaining requirements. The cycle then repeats itself until the user is completely satisfied with the system. However, there is, by and large, no disagreement about the required activities. Thus, all models require that each of the above activities is performed during the software development project.

A little reflection will also show that it is possible to group the various activities. Thus, activities 1 and 2 are primarily concerned with understanding what the software is supposed to do, phases 3 and 4 are primarily concerned with building the system, while phases 5 and 6 concern the actual installation and operation of the system.

A Life Cycle Model for System Acquisition

The life cycle for software acquisition contains many of the activities that are also found in the software development life cycle models. Assuming that a project has been approved and that the decision has been made to acquire a software package, rather than build it in-house, there are essentially two high level activities that have to be performed, namely

1. Software selection and acquisition
2. Software implementation and integration

Software Selection and Acquisition

It will be clear that the main objective of the software selection and acquisition activity is to select the appropriate software package, and to acquire it. It consists of a number of overlapping activities that are often in an iterative manner, including

1. Planning
2. Information Search
3. Selection
4. Evaluation
5. Negotiation
6. Choice

although which activities are included and the attention given to them vary from acquisition project to acquisition project. For example, the effort required, and attention to detail, in a multi-million dollar project to acquire an enterprise system for a large manufacturing organization is significantly different from the effort required in a project to obtain and implement new web browsing software.
Planning.

As in any project, the starting point in the software selection and acquisition phase of a software acquisition and implementation is a series of planning activities.

One of the first activities is the formation of an acquisition team. Often an acquisition project starts with the appointment of a project manager, and it is typically left to the project manager to form an acquisition team. In finding the right people for the acquisition team, the project manager will need to take into account a number of different considerations. For example, he or she will need to make sure that the persons selected have the time (or can find the time) to actively collaborate on the project. Also, complex systems often impact a large number of users within an organization and it is therefore important to get input into the process from as many user representatives as possible. Moreover, the new software will often have to be supported by the existing IT staff, and they therefore have to be involved in the acquisition and implementation project. Also, since in many cases the software to be acquired constitutes a significant investment, it makes sense to have representatives from the accounting and purchasing departments involved as well. Finally, since large software acquisition projects often involve complex legal negotiations, one might want to include representatives from the legal department in the acquisition team as well.

Another activity that takes place during the planning phase is a decision about how to go about doing the selection and acquisition process. For example, it needs to decide which information gathering activities it will perform. Will the team create a Request for Proposals (RFP)? Will it invite vendors to conduct demos, and, if so, how will these demos be performed? Other important consideration concerns the vendors to be considered and criteria for identifying them. For example, some organizations have policies in place that determine whether a particular vendor can be considered or not. Another decision concerns whether to use external consultants. It is often the case that organizations do not have sufficient in-house expertise to be able to fully evaluate different software packages. In such cases, organizations typically turn to external consultants, and this clearly raises the question how these external consultants should be identified and on what basis they should be contracted.

Another important activity that is included in the planning phase is the requirements analysis and definition, and this activity is as important in an acquisition project as it is in a software development project. The acquisition team needs to fully understand on what the scope of the acquisition project is. For example, if the project involves the acquisition of a new PC-based word processing package, the acquisition will need to know exactly which departments will receive the new package and how many users the software needs to support in each department. Or, many large complex software packages, such as enterprise systems, contain a large number of modules, and the acquisition team will need to know which modules are to be included in the acquisition
project. If an organization is not clear about what services it expects the new IT application to deliver, it will be very hard to acquire the appropriate system.

During the planning phase, the acquisition team typically also formulates selection and evaluation criteria. Organizations that are looking to purchase large software systems often issue a so-called RFP (Request for Proposals). RFPs are typically very extensive documents, especially for large complex acquisition projects, and the responses received from potential vendors are long as well. Since the acquisition team is unlikely to have the time to read all responses in detail, it becomes important that the team find a quick way to select those responses that it will evaluate in detail. In order to systematize both the selection and evaluation activities, the acquisition team must decide prior to engaging in these activities on the criteria that it will use to select those responses that it will evaluate in detail and the criteria that it will use in evaluating the responses. Ideally, selection and evaluation criteria are specific prior to the evaluation of any product. Without it, there is a real danger that the team will be impressed by features of the software that, in the end, have very little bearing on what the organization needs.

Since selection criteria are used to quickly reduce the list of potential vendors to a smaller size, selection criteria are typically easy to check. Examples are the years that the vendor has been in business, whether it has a local support office, whether it strictly adhered to the communication protocol that the buyer established, and so on.

In contrast, evaluation criteria are much more involved. For example, during the evaluation phase, the team will typically try to determine, for each requirement identified in the requirements analysis, whether the product to be purchased meets it. Since the number of requirements in some cases may run into the thousands, determining whether a particular vendor or product meets all the requirements becomes a time-consuming process and certainly not one that the acquisition team can afford to go through for every vendor.

Evaluation criteria often have weights attached to them to reflect the fact that the team may consider compliance with one evaluation criterion more important than another. For example, compliance with the requirements will typically have a higher weight than price to reflect the fact that the organization will prefer a slightly more expensive product or service that meets all its requirements than a cheaper one that does not meet all its requirements. However, the organization will also want to put a limit to this. In some cases, the price for the product or service may be so high that the organization may prefer the cheaper one, even though it does not meet all its requirements. Similarly, if a team considers certain requirements to be more important than others, it may attach a higher weight to those requirements. During the evaluation phase, weights are used in a fairly straightforward manner to calculate an overall score for a product. Thus, the team will assign a rating to each evaluation criterion and multiply this by the weight to arrive at a score for that product or service on that criterion. It will
then add the scores on each evaluation criterion to arrive at an overall score for the product or service.

Although selection and evaluation criteria should be formulated and agreed on by the acquisition team prior to any evaluation of a product, they should not be treated as set in stone. For example, during a demo, it may emerge that a software product has features that the acquisition team had not thought about before and that are genuinely useful to the organization. It would be counterproductive for the team not to use these newly acquired insights and for example revise its requirements analysis document, and/or to change its selection and evaluation criteria. However, when such changes take place, it is important that products and vendors that have already been considered be re-considered in the light of these new criteria. After all, it may well be the case that a product that the team already looked at also has the features in question but that the team did not see these features for the simple reason that it did not think to look for them. Reconsidering the products already investigated when the selection or evaluation criteria change increases the chances that the team will purchase the most appropriate software.

Another activity that is an important aspect of the planning activity is a marketplace analysis. The primary objective of this activity is to identify the main vendors capable of providing the IT systems or service that the organization is seeking to acquire. Which companies provide the type of software that the organization is interested in, and what is their market share? What is their financial position and how likely is it that they will disappear in the near future? What other organizations have the vendors supplied software to, and so on?

Information Search

Information Search is clearly an activity that has to take place throughout the selection and acquisition project, and is often an iterative process. Different phases of the acquisition project will require essentially the same information but at different levels of detail. For example, the team will have to obtain information about the vendor for the marketplace analysis that it performs during the planning phase, during the selection phase and during the evaluation phase. However, the level of detail of the information that it needs for each of these activities clearly differs considerably. For example, during the selection phase, the team may simply want to determine whether the vendor has been in existence for at least five years, whereas during the evaluation phase the team may look very carefully at the financial performance of the vendor over the last five years.

There is a cost associated with any information search. Some of these costs are direct. For example, the team may use consultants in its information search or buy reports written by consultancy firms. Some of these costs are indirect and involve the time of team members conducting the information search. Moreover, it is almost always
possible to find additional information about some topic. The challenge that the team faces is to find a sufficient amount of information to make an informed and justifiable decision at the lowest possible cost, and there is a trade-off here that the team has to bear in mind. Obviously, it has to spend enough time and money to find sufficient information to be able to make a good decision while at the same time making sure that it finds only the information necessary to make a good decision. On the one hand, it should avoid spending too much and obtaining information that is irrelevant. On the other hand, it should avoid arriving at a decision too hastily and without considering all the relevant information.

The information search process is itself is relatively simple, and consists of three steps. First, search for the relevant information. Second, screen the information. Third, decide to keep or discard the information.

There are a lot of information sources that an organization can use in its acquisition projects. In general, one can distinguish between internal information sources, such as policies and procedures that the organization has in place, plans and reports that the organization has written, and individuals within the organization. External information sources include the software vendors themselves, industry research reports such as those produced by external research firms, and external consultants.

Another method that acquisition teams often use as they attempt to identify the best product and vendor is issuing a so-called Request for Proposal (RFP). An RFP is a document describing exactly what IT system or service the organization is looking for and exactly what information an interested vendor should provide. Good RFPs are extremely time-consuming to create, and writing an RFP is therefore often an expensive process. However, in general the quality of the information the buying organization can expect to obtain from an RFP is extremely high, and often the response that the vendor submits to the RFP becomes part of the contract that the organization and the vendor will ultimately agree to. Given the importance of RFPs, we will devote a separate module to them.

Acquisition teams will often also use vendor demonstrations as part of their information search activities. Demos are a useful way to gain additional information about the product that the vendor is proposing to sell. However, there are a number of complications that an acquisition team should bear in mind.

The first complication concerns the nature of the demo. There are essentially two options. The first option allows the vendor to take the lead and to demonstrate the product in whichever way the vendor sees fit. The second option is one that forces the vendor to demonstrate how its system would conduct particular tasks, as determined by the buying organization. Prior to the demo the vendor is presented with a set of scenarios and during the demo the vendor will show how its system can be used in those scenarios. For example, if an organization is looking to buy a CRM system, a likely
scenario involves entering a new user. Scenario-driven demonstrations are often called “canned presentations”.

Vendor-driven demonstrations are in general of limited use. A vendor will obviously want to show off its product in the best possible light and may try to limit the demo to the strongest features of its product, and these features may or may not be important to the buyer. On the other hand, asking vendors to demonstrate how their systems perform in certain scenarios requires that the acquisition team formulate such scenarios in the first place, and formulating scenarios is potentially a time-consuming and expensive activity.

Acquisition teams will often use a combination of a canned presentation and a vendor-led demo. Thus, the vendor may be asked to first show how its system performs on the scenarios that the acquisition team identified, and may then be invited to show any other functionalities that it believes may be of use to the buying organization. Demos of this type have the advantage that they both show the buying organization what it wants to see, while at the same time allowing the vendor to demonstrate additional features. Since the vendor is likely to have sold similar systems to other buyers and may even have included features at the request of other buyers, it is possible that the vendor is aware of desirable features that the organization interested in buying the IT system may not have thought of.

Selection and Evaluation

In many ways, the most crucial decisions in an acquisition project are made during the selection and evaluation activities. In many projects, the list of software products and vendors identified during the market analysis conducted in the planning phase is too long in that the acquisition team will simply not have the time to evaluate each in depth. If this is the case, the acquisition team will move into a selection phase, whose primary objective it is to reduce the list of vendors to a manageable size.

Having selected a limited number of products and vendors, the team will go on into an evaluation phase during which the software systems that were identified during the market place analysis and “survived” the selection process are evaluated in depth. It is useful to make three observations about the evaluation process.

First, in most cases, there is no ideal solution. It is extremely unlikely that any software package will obtain a perfect score on all the evaluation criteria that the team has identified.

Second, choosing any IT system is ultimately a cost-benefit issue. The team will need to determine whether the benefits that the software to be acquired and implemented will bestow on the organization outweigh the costs. Moreover, in making this decision the team has to take into account the fact that there are many different costs associated
with an acquisition and implementation project. In addition to the direct costs involved in actually purchasing the system and, if necessary, to hire external consultants to help in the acquisition and implementation, there are many indirect costs. Examples of indirect costs include the time of the members of the acquisition team, the need for training to allow members of the organization to make optimum use of the system, and the cost that the organization will incur because of business interruptions during the implementation of the system. Moreover, in many cases, the acquisition team has to take into account opportunity costs, the fact that the money used for this particular IT acquisition and implementation project could have been used for different projects, which may have a higher pay-off for the organization.

Third, as we indicated a number of times before, evaluation is likely to be an iterative process. As the team evaluates IT systems or services, it may become aware of additional requirements or other additional evaluation criteria. When that happens, products that have already been evaluated will need to be re-evaluated.

*Negotiation and Choice*

Assuming that the acquisition team has settled on a particular vendor and its product, the final activities that take place during the system selection and acquisition phase are the *negotiation* and *choice* activities. There are essentially two types of negotiation, namely business negotiation and legal negotiations. Business negotiations concern questions about the match between the software under consideration and the buying organization. For example, software vendors are often prepared to make changes to their software to better meet the requirements of the buyer. Legal negotiations, on the other hand, cover the legal aspects of the acquisition and result in a contract between the vendor and the buyer. Typically, a buying organization will only start legal negotiations with the vendor that it has identified as providing the best product or service.

The choice and negotiation activities sometimes iterate. For example, it may happen that, during legal negotiations, it becomes clear that the vendor is not willing to meet some legal requirements that the buyer wants to impose. In such cases, it may well be the case that the buyer and vendor cannot come to an agreement and the buyer is forced to consider another vendor, although one can of course argue that an organization that finds itself in a situation in which it cannot come to an agreement with a vendor has not done its requirements definition and analysis as well as it should. After all, if the legal requirement is a deal-breaker, it should probably have been explicitly included in its requirements definition document.

*System Implementation and Integration*

The second major phase in a software acquisition and implementation project is the software implementation and integration phase. Again, as the name suggests, the
primary objectives of this phase are two-fold, namely (i) to implement/install the new software, and (ii) to integrate the new software into the existing organization. Again, the system implementation and integration phase typically involves a set of separate activities, including

1. Planning
2. Analysis and Design
3. Configuration
4. Transition
5. Operations and Maintenance

Planning

The implementation planning phase is a complex process comprising of different activities and components and the phase should therefore begin with planning. Many of the activities in this phase directly follow from the fact that the complexity of the implementation and integration phase, especially for large IT applications, suggests that it is best to treat integration and implementation as a separate project.

One of the first activities in the implementation and integration phase is the assembly of an implementation team. The main purpose of this activity will be clear. Just as the selection and acquisition phase typically required a team, the implementation and integration phase will require a team.

The question may arise why there is a need for a new team at this stage. Why cannot we simply continue with the team that was formed to conduct the selection and acquisition phase?

The reason is fairly simple: Some of the expertise that was crucial during the selection and acquisition phase is no longer need during the implementation and integration phase and, vice versa, the implementation and integration phase requires expertise that we not necessary during the selection and acquisition phase. For example, while the acquisition team may have needed someone who was familiar with the purchasing policies and procedures the organization had in place, such expertise is less likely to be required during the implementation phase. On the other hand, since implementation and integration probably requires some user training, it might make sense to include a training expert in the implementation and integration team.

Although there is a need to form a new team, the implementation and integration team will ideally include at least some members from the selection and acquisition team. An overlap between the two teams may reduce potential conflicts as the implementation and integration team members who participated in the selection and acquisition phase will be able to explain to new members the rationale for the selection of the particular software package that was chosen. However, since the implementation and integration
phase is likely to require a different set of skills than the selection and acquisition phase, it is neither likely nor desirable that all the members of the selection and acquisition team and only those make up the implementation and integration team.

An important element in the planning process is the determination of the scope of the implementation, and the implementation and cut-over strategies to be used.

The question of scope typically arises because large software systems, such as enterprise systems, consist of a large number of modules. For example, SAP (www.sap.org), one of the largest and most widely adopted enterprise systems, lists about 50 categories of applications on its web site, each of them supported by tens of modules. The implementation team will have to make a decision whether all modules should be implemented at once (also known as a “big bang” implementation), and, if not, which of the modules should be implemented, and whether there is a priority order between the modules. A module by module implementation is called a “phased roll-out”. It has the advantage that at least parts of the new system become available sooner, and that, overall, there probably is less interruption of the business. However, since the new modules will in general have to be integrated with legacy systems that have not been replaced yet, a phased roll-out will require the creation of interfaces between these legacy systems and the modules to be implemented. Hence, phased roll-outs are likely to require significantly more work.

Often scope-related issues will have been considered during the selection and acquisition phase. The fact that members of the acquisition team will know the reasons for these decisions shows why it is important that the implementation team includes at least some members of the acquisition team.

Another scope related decision concerns geographical locations. Large enterprises often have multiple locations, and the implementation team may have to decide whether to implement the new software in all locations or only some, and if so, which. One option would be to do big-bang implementation at a single site and use the lessons learned from this implementation and integration project at other company locations.

Scope decisions have to be based on the business drivers for the project. If the business driver is the replacement of one or more legacy applications that the organization no longer can or wants to maintain, then the scope of the implementation and integration project should minimally include those modules that replace those legacy systems. If the primary business driver is to achieve standardization between the various units of a large enterprise, then the team should probably not restrict the implementation geographically. In other words, the new application should be implemented at all locations of the organization, although it may of course achieve this on a location-by-location basis.
A closely related question concerns the cut-over strategy to adopt. It is more than likely that the new IT system will replace some legacy systems. The question therefore arises how to switch from using the old systems to the newly installed system, and the strategy to do this is referred to as a cut-over strategy.

There are a number of possible cut-over strategies that the team can adopt. Perhaps the simplest is a direct cut-over, in which the new system is installed and, at an appropriate time, all older systems are switched off and the organization starts using the new system. In many ways a direct cut-over is the simplest cut-over strategy but it carries certain risks with it. For example, unlike the other cut-over strategies, the new system is never run in parallel with the old systems and it is therefore hard to compare the results from the new system in operation with the results of the old systems in operation. Moreover, a direct cut-over typically requires the business to be interrupted for some period. This option therefore becomes problematic for organizations that cannot afford to have their systems go down, such as airline reservation systems or online banking systems.

The alternative to a direct cut-over is parallel conversion in which the old systems and the new systems are run in parallel for a limited period. The advantage is that this allows for the comparison of the old and the new system.

However, parallel conversion has a number of drawbacks as well. First, running the two systems in parallel puts enormous pressures on the IT infrastructure, especially if the organization has adopted a big bang implementation. After all, both the old and the new system are likely to require significant hardware resources to run. Since most organizations will not have the IT infrastructure that allows them to simply install a large new IT application, such as an enterprise system, running both in parallel will at best result in significant increases in processing time, and, at worst, frequent crashes of the servers etc. Second, running both systems in parallel will essentially double the effort expected of end users. After all, not only will they have to enter data in the old system, they will also have to enter the same data in the new system.

Other forms of parallel conversion counteract at least some of the drawbacks of a full parallel conversion. For example, an organization may adopt a limited parallel conversion in which only some of the legacy systems are retained and run in parallel with the modules in the new system that will eventually replace them. Clearly, this reduced the pressure on the IT infrastructure and will only increase work for a limited number of employees, but it has the drawback that the implementation team may have to write additional interfaces to ensure that the legacy systems that are run in parallel with the newly installed modules interface to the other modules in the new system.

Another option to the organization is retroactive parallel conversion. Retroactive parallel conversion essentially involves running the new system on historical data. It has the advantage that it does not put any added pressures on employees not directly
involved in the implementation project. However, it does put pressure on the IT infrastructure, and still requires a direct cut-over at some stage.

In order to relieve pressure on the existing infrastructure when testing and/or implementing a new IT application, organizations sometimes create sets of independent hardware and network configurations that very similar to each other. Such a configuration is called a landscape, and one can distinguish between different landscapes dependent on the number of so-called tiers, i.e. replications of the hardware and network configuration\(^1\). Thus, a tier-1 landscape only has the production environment; a tier-2 landscape adds a development and testing environment; a tier-3 landscape, which is the most common, separates the development and the testing environment, and therefore has a production environment, a testing environment and a development environment; and a tier-4 landscape adds a pre-production tier between the production and testing environment, which can, for example, be used for training. It should be noted that tiers in a landscape do not have the complete physical replications of the production environment, or even has to reside on different servers. The distinction is conceptual, not physical, and the increased use of virtualization and the availability of virtualization tools make it less cumbersome to establish multiple tiers.

The final set of issues that the implementation team may have to consider during the planning phase is the likely impact of the new system on the existing IT infrastructure. In many cases, the new applications will require the organization to add for example servers to its IT infrastructure. Also, the new application may require the organization to acquire additional software tools and/or applications. For example, the database management system that underlies current applications may not be suitable for the new IT application that the organization is implementing. It is not unusual for organizations to have to rely on outside consultants to help them make decisions such as these. Moreover, IT personnel may need to be trained on the operations and maintenance of the enhanced IT infrastructure.

**Analysis and Design**

As said, large software applications, such as enterprise systems, often have business processes embedded in them, and their successful implementation requires an organization to adopt these processes. Moreover, those systems that are less rigid in the business processes that they expect the system to have in place often require the organization to model its business processes in the software. Either way, an important phase in an IT application implementation and integration projects is analysis and design. The primary objective of this phase is to complete a gap analysis between the

---

\(^1\) Note that this use of the term “tier” is different form the way in which the term is used in the context of n-tier systems., such as 3-tier system. In such systems, each tier provides a different functionality (e.g., a display tier, or a database tier).
structure of the organization and its current business processes and the structure and business processes that it would have to adopt to obtain the greatest benefit from the application it acquired.

We have repeatedly already pointed to the importance of business process modeling in any IT system or service application project. Ideally, an organization has a complete model of its business processes during the selection and acquisition phase, for example as it prepares a RFP. Moreover, ideally organizations will have made some preliminary assessments about how well the business processes in the new system fit the organization’s current processes, or how easy it would be to model the organization’s current business processes in the software to be acquired. However, because of the time involved in completing these tasks fully, organizations can only be expected to have made preliminary assessments. It is now time to make a complete assessment.

The first step in this process is an “as-is” assessment. Essentially, this requires the organization to form as complete a picture as possible of its current structure and processes. Not only does such an “as-is” assessment help in the implementation and integration process, it often also points out issues with the current processes, such as duplication and/or missing processes. For example, in a thorough analysis of the business processes of an agency that provides services to individual customers, it may be discovered that many of the business processes have a common sub-process to locate customers whose contact information is no longer up-to-date, and that all individuals perform this sub-process, even if their primary responsibility lies elsewhere. Having identified this, one can now ask the question whether it might be more efficient to create a separate locating customers process and employ specialists to perform this process.

When modeling business processes, the implementation team will also want to determine what the data and information requirements are for each process, and where this data or information resides. Also, it will try to determine the business rules that govern each process. Thus, in the previous example, the implementation team will want to know what data or information sources are used to locate customers whose contact information is no longer up-to-date, and what the business rules are to govern the process (for example, is there a limit on the number of attempts to locate a customer).

In addition to a thorough analysis of an organization’s business processes, the implementation team also defines the structure of the organization in terms of its organizational hierarchy (divisions, departments, units etc) and its financial organization (cost centers, the basis for cost allocation, chart of accounts, etc). Enterprise systems may make certain assumptions about the structure of an organization, and the team may discover mismatches between the current organizational structure and the structure assumed by the enterprise system. Such issues obviously would need to be resolved. And, even if the new IT application makes no assumptions, the information
gathered during this exercise will prove invaluable in configuring the new application, as we shall see in the next activity.

A second step in the process analysis and design phase is to form a better picture of what the organization would look like after the most effective implementation of the acquired software package. In other words, the implementation team tries to form a picture of the organization “as should be”. This activity will typically be informed by the information gathered during the selection and acquisition process (e.g., requirements, expectations of the software, and so on).

In creating a model of the organization “as should be”, the team needs to develop an image of the processes embedded in the application that was acquired. In many cases, the software installation manual or other support material provided by the vendor allows an organization to do so. However, in many cases, the support material is insufficient and it is not unusual for an organization to require the assistance of consultants who are familiar with the application in this activity.

Another technique that is often used to help in the analysis of the organization “as should be” is simulation. Essentially, simulation involves making an initial configuration of the software, asking a number of key users, who are often members of the implementation team to run the software and to identify issues. Examples of issues that may be identified in this way may be the fact that the software deals with data in an unexpected way, for example by flagging a particular case unexpectedly as requiring special attention. Or, the team may discover that the software provides some unexpected functionality that could be used to automate certain processes that are now done manually. Alternatively, the team may discover that the software does not provide some functionality that the organization currently relies on. Finally, the team may discover that there are discrepancies between the processes embedded in the software and the processes currently implemented in the organization.

Having completed an assessment of the organization “as is” and the organization “as should be”, the implementation team can now identify mismatches between the two, and decide how it is going to deal with each. The team has several options open to it. For example, it can accept the process embedded in the software and modify the organization’s business processes. Alternatively, it can accept the organization’s business processes and adapt the software.

There are a number of ways in which the software can be adapted to the organization. For example, in cases where it turns out that the software allows the organization to automate certain processes, it can decide to stick with the manual process. In cases where the software does not support current business processes, the team can ask itself whether such processes are indeed necessary or whether the way in which the software achieves the effect of the missing process are preferable. For example, a software application may not support the creation of customer records by customer support
representatives as it supports self-registration by customers over the Internet, and has an embedded business rule that customers must register themselves. Clearly, if such a business rule is not acceptable to the organization, the implementation team has two options. If it already has an application in place that supports the customer registration function, then it can decide to try to integrate this application and the acquired software by writing an interface between the existing application and the acquired application. In cases where it does not already have such a system in place, it can decide to create a so-called “bolt-on” to the software application to also allow customer representatives to create customer records.

While the implementation team will in general have to use its professional judgment, its default position ought to be not to adapt the software to the organization but to adapt the organization to the software. The reasons are related to the reasons for buying, rather than building, software discussed in an earlier module.

First, customizing software, creating an interface between an existing application and the new software, or creating a bolt-on is essentially a software development project, and such project are notorious in being over time, over budget or not delivering on the specified functionality. In other words, deciding to adapt the software to the organization by writing additional software is likely to have a negative impact on the implementation project both in terms of cost and schedule.

Second, software applications are frequently updated for example because the software vendor fixes bugs or adds functionality. However, if the organization has customized the software or has created an interface or a bolt-on, there is a chance that the customization, interface, or bolt-on will not work in the upgraded software, requiring yet another software development project and delaying the time that it will take to adopt the upgraded version of the software.

Finally, large software packages typically evolved out of bespoke applications that were developed for world-class companies. It therefore seems safe to assume that they embed globally best business practices and processes. Indeed, one of the reasons for the acquisition of well-established enterprise systems is that they allow the buying organization to take advantage of the business processes that have proven successful in the best firms in the world. By adapting the software to its current business processes, the organization runs the risk that non-optimal business processes will persist.

The latter argument also points to the best reason an implementation team can have for deciding to change the software, rather than the organization. If the organization is convinced that its current business processes give it a competitive advantage over its competitors, then changing business processes is clearly disadvantageous. This typically is the case when the firm operates in an environment that is significantly different from the environment that the software was developed for. For example, logistics software that was developed for one of the best logistics companies in the world is likely to
assume a reliable transportation infrastructure. After all, it is more than likely that the best logistics companies are located in either the US or Europe and both areas have a good road infrastructure, (relatively) reliable train networks and so on. However, such assumptions are less likely to be true in low-income countries in, say, Africa and it may well be the case that the business processes that are successful for a large US or European logistics company are less successful in a low-income country with a less reliable transportation infrastructure.

Configuration

An important step in any large IT application implementation and integration project is configuration. Most large software systems support multiple versions of a business process and include switches in the software to determine which version of a process the system will support. For example, the software may support a version of the purchasing process that requires each purchase to be signed off by a supervisor and one in which purchases below a certain limit do not need any additional approvals. The team has to decide which version of the process it wants to implement.

In addition to setting switches, configuration often also requires the implementation team to populate a set of system tables. For example, many large organizations divide their operations along regional lines (e.g., North American operations, European operations, and so on) and most large software applications support this. However, most large software applications will not have a fixed geographical divisions built into them for the simple reason that different organizations divide their regional operations differently. Instead, they include a regional division system table that, once populated, will tell the software the regional division along which the enterprise is organized.

A further issue that may have to be resolved during the configuration phase is the integration of the new system with legacy systems that the organization has decided to retain. Typically, modules within the new system will rely on information provided by other modules, and typically they will assume that the other modules have been implemented as well. If the organization decides not to implement these other modules, but to rely on retained legacy systems to provide such information, the IT application needs to be configured so that it obtains the relevant information from the right place.

Since both customization and configuration essentially involve adapting the software to the organization, the question may arise what the difference is. After all, the previous section gave a set of arguments against customization while this section states that configuration is an essential part of any implementation and integration project. The difference is that configuration is an adaptation of the software to the organization within the boundaries allowed by the software. In other words, one does not have to make any changes to the software itself in order to configure it. The adaptations are
supported by the software. This is not the case for customization. In the case of customization, the software itself has to be changed.

While it is fairly straightforward to describe the tasks that need to be performed during the customization phase, the phase itself is often complicated and time-consuming. In an earlier module we said that one of the trade-offs between building software and buying software is that a software development project typically consumes most of its time and resources in the development phase while implementation and integration is relatively straightforward, while in the case of a software acquisition project the selection and acquisition phase is considerably less time-consuming than the implementation and integration phase. One of the reasons that the implementation and integration phase is so time-consuming in an IT system acquisition project is the sheer number of configuration decisions that have to be made and realized.

**Transition**

The primary objective of the transition phase is to install the new IT system and to start using it. Essentially, it involves putting the plans created in the previous phases into practice. It consists of a number of activities.

The first activity is the creation of the necessary additional software. Although we argued that in general an implementation team’s attitude ought to be to avoid customization, there may be situations in which customizations are desirable. Similarly, the special business environment that the organization finds itself in may require the creation of bolt-ons. Obviously, such software has to be developed.

Another class of software applications that may have to be developed during the transition phase is interfaces to legacy systems that the organization has decided to retain. Finally, since the organization is likely to have valuable information in its legacy systems that it wants to be converted to the new application, the implementation team may have to create some data conversion software. Generally, large software applications will come with a set of tools that makes the creation of interfaces or data conversion programs more straightforward.

Data conversion may itself throw up a lot of issues that need to be resolved. First, there is the issue of how to convert data that is highly dynamic. Consider for example an online banking application. Such a system contains both fairly static data that changes relatively infrequently, such as account holders address information, and highly dynamic data that changes very often, such as account balances. While it is fairly straightforward to convert static data, it is much harder to convert dynamic data, especially in cases where the organization does not want to bring down its old systems while it is installing its new systems. Many of the techniques that can be used in mirroring web systems prove useful here as well.
A second issue that often arises during data conversion is the existence of inconsistent information. For example, a university may be implementing an enterprise system in which each individual, no matter what their status at the university (student, faculty and staff, alumni, etc) has a single record in a person data base. It is not inconceivable that the university had separate systems in place to keep track of students, faculty and staff, and alumni. Moreover, it is likely that at least some of its alumni are now employed by the university, and that there are two records for such individuals, one in the alumni database and one in the faculty/staff data base. Keeping the same piece of information in two separate locations is of course a potential source of inconsistencies in that the value recorded in one location is inconsistent with the value recorded in a different location, and avoiding such inconsistencies is in fact one of the primary reasons that relational databases are normalized. However, since we are consolidating information from different databases into a single one, we may discover inconsistencies, which need to be resolved.

Another data conversion related issue is the fact that the new IT application may require data that were not captured originally. Again consider the example of a university enterprise system, and consider the case of transfer students, students who have completed part of their degree requirements at another university before transferring to this university. The legacy system may have recorded that the students has completed a course elsewhere, but may not have recorded at which institution. The new enterprise system may also support the recording of this information. Since this information is missing for existing records, a requirement that such information be included in the new system would lead to a set of additional issues.

The transition phase will also include testing. There are two types of testing that are relevant in this context, namely unit testing and integration testing. Unit testing concerns individual modules; integration testing concerns testing the units in conjunction with other units in the system. Typically, testing will involve the business experts. Also, it is important that testing be done on realistic amounts of data, especially if response time is an issue. Testing a program on a database with 100 records will tell the team little if the production database is likely to contain over 100,000 records.

A further activity that needs to take place during the transition phase is the detailed planning and delivery of training, both of the eventual end users and the IT support personnel. An obvious, but often overlooked, point is that providing training is not sufficient; one must also ensure that those who need the training will actually receive it. There are some cases where organizations required personnel to complete training programs outside normal working hours and without offering any additional compensation. In such cases, it is not unlikely that personnel either will not participate in the training at all, or, if they do, only make a half-hearted effort to grasp the relevant material.
The final activity in the transition phase is to implement the redesigned business processes and to cut-over to the new system.

Operations and Maintenance

Once the new IT application has been implemented and integrated, it has to be operated and maintained. An important element of this phase is to provision of continued user support.

Another possible post-implementation task is a post-implementation review. Any review of a project is likely to be beneficial to an organization, both because it can identify lessons that it can in future projects, and because it allows the organization to put measures in place that may help it resolve any remaining issues. IT system and service acquisition projects are no exception. Essentially, the objective of any post-implementation review is to determine to what extent the implementation has been successful and has delivered the expected benefits. If not, then the organization will want to identify the reasons and find and implement measures to resolve the problems. For example, the organization may discover that end users are not making use of a particular module. It may then determine whether this is the case because users are not aware of the module, because users do not believe that the module improves the efficiency and effectiveness with which they discharge their duties, or because the users are deliberately sabotaging the organization. Clearly, whatever measures the organization puts in place to resolve the issue of users not utilizing a particular module fully depends on the underlying cause. Any post-implementation review will probably also want to investigate how well the new system fits into the IT infrastructure.

Conclusion

In the module we discussed the rationale for life cycle models in general and a life cycle model for system acquisition projects in particular. It has to be added that, while life cycle models have been studied extensively in the context of software development, they are less well established in the case of software acquisition. However, whether an organization adopts the life cycle model discussed in this module, or whether it adopts some other methodology to manage the software acquisition and implementation process, it is crucial that the various issues discussed in this module are considered.