BSS transformation

Approach for predictable business transformation including BSS renewal or replacement

June 2021

RA Advisory

Replacing core applications in industries with high level of IT dependencies is a challenging proposition with high failure rates. This document presents a categorization of approaches and a more detailed description of a structured, proven approach to greenfield replacement of telecom BSS[1].

The detailed approach described is focused on leveraging vendor capabilities to achieve competent execution and risk management, as well as simplifying business through the use of standard systems. In this way, the typically much-needed product and process simplification become part of the core process. So, while the mechanics of the approach is focused on implementing systems, it includes a business transformation in part as a prerequisite and in part through the process.

The approach focuses on cost predictability, both in the project execution and in the subsequent operation and emphasizes the need for change management.

---

1 Introduction

Increasing cost and complexity of any non-trivial IT landscape appears to be a natural force that universally drives all companies towards a situation where "legacy" systems are a constant barrier to business agility. Replacement of such "legacy" systems in a predictable way is the topic of this document.

The document is primarily inspired by projects in the telecommunication industry and the examples are taken from there, but the discussion of approaches is applicable in other industries that are highly IT dependent and where a sufficient mature vendor landscape exists. The approach for greenfield replacement requires that there is a number of vendors that can support the core business with a standard system, leading to an attractive competitive situation from the perspective of the buyer. This is not the case in all industries, either because they do not exist or because there are too few of them to create proper competition.

The belief that simplicity and standardization of the transaction processes are of key importance permeates the document. This is fostered by a belief that customers value the stability, ability to interact directly with processes and simplicity of offerings, communications etc. a lot more than the features that may in some cases be missed out from the non-standard systems. And that the lower structural costs resulting from use of standard systems provide other advantages.

Use of custom solutions are viable in selected areas where the value of flexibility and speed can justify the higher complexity and cost of custom systems.[2]

The document is structured as follows.

Section 2 discusses the need for management buy-in, the top-level change management.

Section 3 outlines the alternative approaches for BSS replacement. The rest of the document is focused materially on the greenfield approach.

Section 4 describes selected elements of business transformation pertinent to the greenfield approach.

Section 5 describes the first, preparatory phase of the greenfield approach where the requirements and contracts are delivered.

Section 6 describes the second phase of the greenfield approach, the implementation.

starts with outlining the different approaches for system replacement and continues with a more detailed discussion of a proven approach to greenfield replacement.

2 The imperative for change

There are two elements of change management pertinent to projects as those contemplated here. One is the normal, operational change management of any IT system, preparing people for new processes, tools through training, understanding of shift in responsibilities etc. This is discussed further in section 6.3.

---

[1] BSS = business support system, the name typically applied for the transaction systems in telecom (as opposed to OSS which are closer to the physical network).

[2] These considerations refer to the normal situation of network operators. For organizations where a small team, say less than 20 people, can develop and maintain the entire system stack, custom development can be a viable strategy.

The other and more important is the top management recognition of the imperative for change and its involvement in driving the transformation project. This is required for projects that are transformational since the resistance to the changes, the priorities required etc. must be addressed at top management level.

The organisation needs to understand that the execution of the project is mandatory, and the management team must be motivated by implementing the change and the corresponding transformation. Furthermore, the management team (and potential board and owner stakeholders) must be committed to ruthless execution, including taking out products and corresponding revenue that drives excessive complexity or precludes using standard systems.

This also requires the management team to agree on what 'business as usual' activities shall be postponed or abandoned. Getting such understanding and acceptance in place frequently requires an analytical basis for what must be achieved, essentially parts of a business strategy, as well as softer motivations. And ultimately replacing staff that cannot see themselves motivated by the change journey.

This first layer of change management is one of the reasons for the frequently expressed need for senior management involvement in complex IT projects.

An important discussion, also related to this first layer of change management, is where in the organisation the project refers to. The type of projects discussed here, if they are to have any chance of being successful, will "take over" substantial proportions of management as illustrated in Figure 1: *project overlap with line organisation* below with generic organisational units (IT, products, markets, finance):

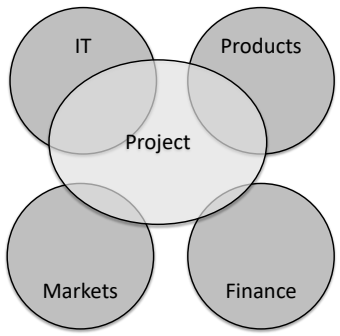

Figure 1: project overlap with line organisation

The project will define significant proportion of the future business: IT, processes, products with impact on revenue, organisation etc. This means that the scope of the line management will be reduced for the duration of the project, potentially significantly. This particular challenge sometimes can result in destructive turf wars.

For project reference, there are generally two main scenarios:

1. The project referring to the CEO.
2. The project referring to the CIO (or CTIO as the case may be).

Both scenarios have the effect of the project "trespassing" into the responsibilities of the line organisation, but reference to the CEO makes it more extensive (since the CIO is the one most affected; the CIO will be responsible for managing the "legacy", including keeping staff motivated for that while someone else builds the new and interesting IT platforms). For truly transformative projects, this is the best approach, but needs the active support of the CIO. In this case, the project manager should be part of the management team in order that the conflicts of priorities, overlapping responsibilities etc. are addressed.

Reporting to the CIO can work in case of the change being less transformational, e.g., a system renewal and incremental process improvements, but still requires a strong, respected and very business-oriented CIO.

Having this basis of top management commitment and understanding of implication is a necessity to secure the stamina required to see through substantial implementation projects.

3 Categorization of approaches

While any number of variations are possible, most are variations of the following:

1. Brownfield, where parts of the system landscape is replaced with a new system.
2. Carve-out, where parts of the system landscape are moved gradually to new components.
3. Greenfield, where substantial portions of the system landscape is replaced with a new system built initially alongside the legacy system stack.

Each approach is described briefly in the following. The greenfield approach is described in a lot more detail; whereas the brownfield and carve-out approaches are very context dependent, the greenfield approach has a large proportion of generic activities.

3.1 Brownfield approach

In the brownfield approach, parts of the legacy landscape is replaced with a new system. Typically, this is a replacement of a CRM, online or billing

system, but can also be smaller components like mediation, output management etc.

The approach is to essentially replace an existing component with a new one, supporting the same set of products as previously.

For small components, this is fine, as these generally do not carry the full complexity of the product portfolio and has comparatively simple and often standardized interfaces.

For larger components, the brownfield approach is more challenging. Firstly, it frequently requires implementing a lot of interfaces, frequently hundreds. Secondly, it requires that all existing products are supported in the new system.

Brownfield may be a viable approach in cases where the inherited complexity can be managed from both a product and an interface perspective.

The product perspective includes limiting the product support of the new system to only the newest generation(s) of product, leaving legacy products to be serviced "somewhere else". For instance, when replacing CRM, the billing system may have rudimentary operational CRM capabilities that may be used for legacy products while the new CRM is used for newer products.

The interface perspective reflects the number and complexity of interfaces affected. Simplicity of interfaces would normally require that the system stack is structured in reasonable layers with well-defined and centralized API separating the various layers.

If these conditions are not in place the brownfield approach should be viewed with scepticism.

3.2 Carve-out approach

In the carve-out approach, functionality in the legacy systems is replaced by moving it gradually to new components. The approach resembles the brownfield in the sense that replacement is partial, and a lot of interfaces need to be built. In the best applications of the approach, the new components only support a subset of offerings, ensuring a clean-up in the process. Transferring all legacy complexity gives the same problems as the corresponding approach for brownfield replacement.

Replacement of components can secure reduced impact through back-propagating updates to legacy components, causing new and legacy components to contain the same data and thus reducing impact on surrounding systems. Since upholding legacy components defies the purpose of replacement, this is a migration approach, not viable for longer term. For spreading the impact of changes, both in risk and cost, out in time, it can be an effective migration strategy.

The approach can, dependent upon the specific application landscape, have the advantage of providing partial results during the process. This is an important risk-reducing factor as the stamina required for complex system replacement can be challenging to uphold. As a gradual approach, it can also reduce the IT risk since components go into production gradually.

The key challenge with the approach is to find a good sequence to carve out. If the carve-out becomes too complex for each component, it will have the challenges of brownfield. If the number of components required to replace to make integration manageable is very large, it will effective be akin to greenfield.

Since the components to be replaced often will not adhere to the boundaries of standard systems, the approach can require quite substantial amount of custom development, not all of which is temporary. This can significantly reduce the value of the replacement, since complexity is more likely to increase gradually in a custom system.

3.3 Greenfield approach

The last options considered here is the greenfield approach. Since this approach is the most generic where a general methodology can be applied, it is described in a lot more detail below.

Briefly, in the greenfield approach, a new stack is built alongside the old stack with suitable integration points allowing for dual operation. Following the setup of the new stack, customers are migrated into it, more or less aggressively.

A key assumption for the greenfield approach as outlined here is that cost predictability is a key consideration. This is defining for the approach outlined below, and means, among other things, that the contracting is targeting a fixed price for the scope agreed.

4 Business transformation

This section outlines items of the business transformation. Some are separate, preparatory steps, while others are considerations on steps that are integral to the core process. The purpose is not to provide a comprehensive guide to business transformation, but to highlight aspects specific to the type of transformation contemplated here, as well as a few common pitfalls.

4.1 Focus on simplicity

Simplicity cannot be emphasized sufficiently; empirical evidence shows that length of operation and simplicity of product are the only significant explanatory factors impacting IT costs for mobile operators. While not directly documented in benchmarks, it appears obvious that complexity in IT and complexity in business is a vicious circle. Complex business requirements drive complex IT systems; which drives long development times; which causes short term solutions that tend to live very long; which causes process faults and manual workarounds.

The agenda of simplicity should therefore focus on making things simple and keeping them simple. Business transformation, therefore, also involves a governance of IT that can facilitate this.

4.2 Start from scratch

To make sure that all current customers are served with relevant products, one need to look at the current products. However, if one starts with current products and processes, one gets stuck in current functionality and complexity.

So, to achieve simplicity, the input should be the main segments, products in the market serving those and from there derive the products to offer in the future. In other words, a process like one would do in a new company.

Once this has been developed over a few iterations, the current product portfolio can be used to check if there are important features or whole products that have been left out.

4.3 Start with the hard parts

While transformations of the kind contemplated here are not commonplace and few, if any, have been executed to a point where one may reasonably call them successful, many suffer from the mistake of starting with "simple B2C" as a basis for the future system stack.

There are two problems associated with this approach. Firstly, building a business model of products, processes, capabilities, and platforms to serve the simple problem will almost invariably fail to accommodate the complex problem. There may be good reasons for wanting results as soon as possible, and given the typical duration of BSS replacement projects, this is natural.

However, to get a robust platform, it is important to start with the difficult customers; the large accounts and public accounts who have IN and converged products; hierarchical and split billing with specific formatting requirements; extensive self-service portals; special handset rules which must be observed in the physical distribution; and so on.

Also, the prevalent mental image in most employees of an MNO is the "no frill B2C" customer, which makes it double dangerous to start with that as many project members will not have an intuitive understanding of the real complexity.

Therefore, in case of a required early release with simpler functionality, the requirements and agreements should include the more complex. This will reduce, but not completely remove, the risk.

4.4 Understand value creation of IT

Coming down to essentials, IT systems of the type discussed here are put in place to automate a process in the business for efficiency, decision making, self-service etc. And the automated process serves a purpose related to business goals, which is why the IT systems make sense in the first place.

More specifically, the value creation spectrum ranges from raw automation over differentiation through efficiency of use to differentiation in understanding and interacting with the customer.

An example of raw automation is mediation[3]. No customer ever sees it, but it must be in place as an interface between the network and the billing platform. And due to the volumes, there is no practical manual alternative.

An example of differentiation via the customer interaction is the web. Having high quality self-service in terms of "visual friendliness", ease of use, quality of processes etc. can make significant difference in the customer perception.

The value of IT should be linked to the places in the value chain that are differentiating, i.e., the places the customers see and value. In some areas like billing, flexible systems are available that makes this easy. In other areas, compromise on the use of standard systems may need to be made. When making such compromises, it is important to understand the full cost impact of custom developed systems, one that is typically underestimated.

In addition to having impact on the system choices and architecture, the view on IT value has impact on the operating model: pure efficiency through

[3] Mediation refers to the process of collecting, reformatting and aggregating information from the core network elements to make them ready for processing in billing etc.

use of standard systems is a more obvious outsourcing candidate than the differentiating web channel.

Discussing and making this understanding explicit as part of the transformation formulation is important as it will help guide the subsequent project in selecting both systems, customization levels and operating model.

Furthermore, if combined a priori with explicit accepted cost levels and put as requirements to the projects, it also gives a foundation for managing the future cost levels of IT in absolute terms. Obviously, most projects have a very explicit relationship to the implementation costs, but the subsequent costs are often managed with a lot less rigor.

4.5 It is a two-way street

Laying the foundation for business transformation and simplicity is a separate piece of work as one needs to design a new business.

However, a key to simplicity is to use standard systems. Such systems come with version 20+ of data models and process support with input from many implementations. There is no way a similar quality can be achieved in a design made from scratch since the required experience cannot realistically be brought to bear in a specific project. For a standard system, the value deteriorates quickly if extensive customizations are made: the ability to follow the upgrade path, utilize new features and avoid own development will be lost.

It is therefore essential to adapt to the abilities to standard systems, unless the value creation of not doing so is clear and the consequences in terms of cost, operating model etc., is understood and accepted.

Furthermore, the requirements specified in the initial phase of business transformation should recognize this fact and try to keep a fairly high level of abstraction as well as focusing on differentiating requirements; for instance, the ability to rate and SMS or produce a basic bill is not likely to be missing in any system today, whereas hierarchical split billing handling the relevant tax quirks may not be standard.

One reason that the approach outlined here advocates starting from scratch is to facilitate this two-way thinking: define the products with which to serve the customers and use that, rather than existing functionality, as the starting point. And in doing so, bear in mind that product is a lot more than raw telecom functionality and price. Product is also how the customer experience the processes (and, implicitly, data), e.g., through flexibility and usability of self service.

In the approach outlined in section 5 and 6 of this document, the two-way street view becomes very explicit.

4.6 Flexibility requirements

While simplicity is important, it must be kept in perspective; the end goal is to serve relevant customer segments. If a tender is received that can shift market share visibly, specifying a mandatory bill format, which requires a tweak to the billing system, then the billing system will be tweaked.

This observation has three implications:

1. Flexibility must be in place where it matters, particularly in the customer facing parts of the systems.
2. Care must be taken to specify the flexibility requirements as differentiating requirements.
3. A robust IT governance must be put in place to secure that only crucial changes are implemented this way and the resulting technical debt is kept under control.

4.7 Staffing

It may appear odd to single out staffing at this level of discussion. However, staffing the business side of major projects is, almost invariably a challenge in major transformation projects. For an approach as outlined above, the issue becomes paramount. Focus here is on the business staff; that does not mean that IT is irrelevant, but typically IT staff is organized to work in projects, making it easier to allocate them. Furthermore, the IT staff is substantially augmented by the vendors providing the software etc.

As discussed above, the approach outlined in this document attempts to define a “new operation”. This entails a very large number of micro-decisions on what are acceptable compromises to support the chosen segments. These decisions will have significant impact on the enterprise value following the transformation as it may shift market share and revenues by magnitudes beyond normal evolutionary development of business.

Therefore, the requirements cannot be specified and approved in “steering group approval” style, as is the normal mode of involving senior decision makers. It requires that these senior decision makers are directly involved in these micro-decisions; if not CxO level, then experienced people 1-2 layers below.

Freeing up such people from daily operation is generally impossible. They are invariably indispensable to delivering the results in the next few quarters.

One potential solution to this is to strengthen these line organisations several months before the project is to start, so that the senior people in effect has been backfilled and can be freed up for the project.

5 Greenfield phase 1: contracting

This section defines a methodology for the first phase of the greenfield approach. The first phase seeks to shape the project and handles the process up to the point where a contract is signed. The contract is the main physical manifestation of the phase, but following the process ensures that the contract enables a simplified, standard-based business support.

The desired characteristics of the approach include:

1. The process should be designed to ensure business and system simplification.
2. The process should be designed to maximize the use of standard systems.
3. The process should secure short- and long-term cost efficiency and commercial leverage.
4. The process should not require a detailed specification in order to start the procurement process.

The is contracting approach section contains the following subsections that are part of or background to the approach.

1. The 'specification trap' and its implications are outlined. Avoiding this is defining for the suggested approach.
2. The target operating model and its role in securing continued commercial leverage.
3. Considerations for cross-border synergies.
4. Vendor lock-in.
5. Potential uses of system integrators.
6. Definition and importance of standard systems.
7. Identification of potential vendors.
8. Methodology for the contracting phase.

The in-project concept of people change management is described in the second part of the approach, in section 6.3; the framing of the project including securing the organisational stamina for execution, is discussed in section 2.

5.1 The specification trap

Before diving into the specification trap, a note on the relationship between a vendor and a customer. The reason is that not infrequently, the notion of 'partnering with the vendor' comes up as an alternative to running a sourcing process. Vendors often advance this notion.

The perspective here is that the interests of a vendor and a customer are fundamentally opposed. The relationship can be fine, constructive, beneficial for both and even friendly. But coming down to essentials, the customer has an interest in getting as many services for as small a cost as possible. And the vendor has the opposite interest. This recognition lies at the heart of the process outlined here.

This is not to say that partnerships do not exist. One real-life example is an Asian mobile operator who agreed with a RAN vendor to pay a percentage of revenue for implementation and operation of the RAN. In this way, the vendor's income is linked to the results of the customer and the opposing interests are less pronounced. But such deals are not commonplace and almost unheard of in the IT system space.

In a 'traditional' waterfall approach to system procurement, the RFQ/RFP processes would attempt to specify all requirements in detail, sometimes in fairly extreme detail. This approach is not employed frequently anymore but serves as illustration of the 'specification trap'. The principle for specifying requirements in this way is illustrated below:

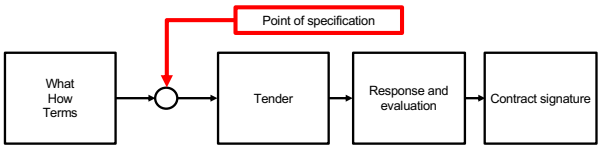

Figure 2: traditional point of specification

The 'point of specification' refers to the point in the process where the specification of the requirements are fixed. The process above is intuitively fine: the customer asks the vendors for a specific solution, vendors respond, and the best fit is chosen.

There are two important problems with this. The smaller problem is that it does not facilitate use of standard systems. The vendor signs up to specifications irrespectively of whether it is standard or not.

The larger problem is that it is not in practice possible. The amount of detail that needs to be written down is prohibitive. It is never right in the first place and it is quickly outdated, sometimes even at the time of submission.

In addition, it is wasteful since one must specify requirements so standard that they are trivially fulfilled by any competent vendor.

An alternative model is to engage with a vendor, typically a system integrator, and run the process as follows:

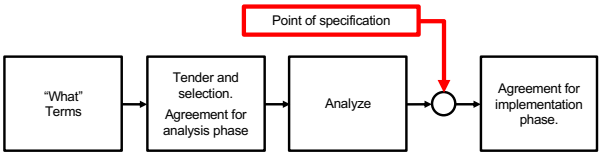

Figure 3: point of specification after contracting

Here, the vendor is selected before the point of specification and assists in the analysis that leads to an agreement for the implementation. From a content perspective the model is fine, but it causes the customer to lose commercial leverage almost immediately. Essentially, it becomes a time and material agreement, at least as far as the system integrator goes. See further below in section 5.5 on the topic of system integrators.

Some vendors would argue that the answer to the dilemma lies in an 'agile' approach. While there are many advantages with an 'agile' approach, from a commercial perspective it still boils down to payment based on time and material. The 'point of specification' is therefore merely dragged out, but still after contracting.

Avoiding time and material contracts is one of the considerations set out above. Fixed-price contracts does not necessarily give cost savings. But they provide cost predictability.

More importantly, time and material projects lack the tension of keeping scope, keeping to standard, focusing on finishing in time etc. which in fixed-price contracts are driven strongly by vendors.

Also, the fixed price contracts ensures that vendors activate their internal risk management apparatus. This is an important assurance when embarking on such a project. In time and material, the risk is normally all with the customer and the vendors will be less concerned with risk management – if the project runs longer, the vendor will gain more revenue.

5.2 Target operating model

One aspiration for the approach outlined is to uphold commercial leverage with the customer. Clearly, it will be stronger before contract signature, but it is possible to keep it throughout the relationship.

The key to securing this is in the operating model following implementation. The primary software vendor will in practice have monopoly for the software. This is due to extreme shifting costs, as shifting to another software vendor entails yet another program like what is outlined here. Therefore, the future relationship with the software vendor should be characterized as follows.

1. The "monopoly" part, i.e., the continued maintenance of the core application, should be subject to fixed, predictable prices that are valid for perpetuity.
2. All other parts should be managed within the customer organisation, or subject to tender. This requires that the competences are not monopolized.

For these reasons, the vendor of software should, following acceptance of the primary delivery, be confined to delivering support and maintenance services. The other services required to operate and develop the solution should be in-house or sourced to other vendors. This requires that competences are built with the customer throughout the project and that sufficient IPR is vested with the customer to permit maintenance of all relevant items. This way, the 'monopoly' component from the primary vendor is contained commercially and the other services can be subject to continued tendering.

The approach above is only possible if the solution remains a standard system as described in section 5.7 below.

The alternative, where the primary vendor continues to deliver development and maintenance services, extends the monopoly to areas where the cost cannot be pre-agreed.

5.3 Maintenance agreements

In section 5.2 above, it was stated that the maintenance agreement should have prices that are valid for perpetuity. This section expands on this and a few other key aspects of the support and maintenance agreements.

Most software vendors will attempt to argue that the termination notices should be "balanced", by which they imply equal timeline. However, equal timeline is nowhere near balanced. For the customer, the time from decision to replacement is a minimum of four years, frequently longer. And the old system is likely to linger a bit longer for 'edge cases'. For the software vendor, it is a loss of revenue with no operational impact.

In practice, permitting a termination notice of less than eight years reduces long term cost predictability significantly. The eight years come are the sum of implementation and decision cycle. The implementation is typically at least four years. The decision cycle is the time it takes to start an

implementation program – typically, the fact that a vendor raises prices will not shift internal priorities to system replacement quickly. Conversely, there is no reasonable business reason to have termination period from the customer longer than a year.

A second point on the support and maintenance agreement is to be careful with upgrades. Firstly, they should be included. Otherwise, the price control of the monopoly will, again, be eroded. Secondly, the assurance that all functionality is maintained in the upgrades, irrespective of the way the system is packaged, should be included.

A final point is to secure a "light-weight" version of the support and maintenance agreement following the decommissioning of the software system. Typically, the benefits of this will be with someone else than the people implementing the system, so it tends to get less attention. But retention of records can be a real pain without having a system to do it with, not made less with GDPR requirements. On the other hand, one does not wish to pay full support for a system with one or two users, to which access is mostly theoretical.

5.4 Cross border synergies

Cross border synergies are discussed here as they, if desired, should be designed into the contract.

The discussion of cross-border synergies is similar to IT outsourcing in that it has an appealing logic ("why do we want to maintain X system stacks when we can make do with one"), which implicitly assumes a context of commonality that is typically not present. Furthermore, also similar to outsourcing, it is not supported by empirical evidence: scale in itself has limited value.

In most situations, the governance of products and processes is decentralized and the requirements to IT are therefore not co-ordinated; and not infrequently there are real market differences, especially for large telecom operator groups with operations in very different geographies. Centralizing the IT people servicing a number of different environments does not give significant scale advantages.

Only in situations similar to where outsourcing makes sense can cross-border synergies provide significant advantages: with high degree of similarity through use of standard systems or centralization of governance of requirements, are there significant advantages of having joint systems across borders. For this reason, the back-end systems like billing, mediation and rating are easier to obtain cross-border synergies for than customer facing systems like online.

In case the new system should support multiple business units, the governance of requirements, releases and relationship to other systems should be in place early, preferably even before embarking on the contracting approach.

The question of cross-border synergies is the topic of another white paper available on the web page set out at the end of this document.

5.5 Vendor lock-in

The approach outlined here attempts to leverage vendor competences in a number of ways, including maximizing the scope of the vendor. If successful, the vendor will have a large footprint in the application architecture. This fact sometimes leads to concerns over vendor lock-in. This section discusses this concern and how it may be dealt with.

Firstly, vendor lock-in is unavoidable (unless one builds systems internally, in which case the 'lock-in' is transferred to critical employees which usually is worse). The lock-in does not become much worse from a large footprint. Any vendor with a substantial footprint is a challenge to replace, typically requiring 3-4 years from decision to execution. Therefore, vendor lock-in must be managed, not avoided.

The approach suggested here is to secure that the areas where the vendor has monopoly is limited to areas where the cost can be predictable. Specifically, securing that all work except maintenance and support of the standard software is done by someone else, including configuration, application operation etc. This is described further in section 5.2.

To avoid monopoly for complex projects where external expertise on running the projects with the chosen system, it is advantageous to select a software vendor that has a network of system integrators familiar with the system. This point is slightly less important than avoiding monopoly on the daily operation, as large complex projects following the main replacement typically are limited in number.

The approach outlined does not remove vendor lock-in; the view taken here is that such a thing is not possible. However, most of the serious issues related to vendor lock-in can be avoided this way.

5.6 Invite system integrators?

Another initial consideration is whether an independent system integrator should have a role in the process. When deciding this, it is important to recognize that there are two potential roles for system

integrators: helping to control the process and assisting with the actual implementation.

There are two scenarios for this discussion.

The **first scenario** is where the new IT system exclusively or materially is delivered by one software vendor that takes a role in implementing the core software directly.

In this case, there are two potential roles for the system integrator. The first role is to run the process outlined here in helping to write the requirements, contracts etc., essentially extending the customer staff. The role can continue into a program management role, still representing the customer side. This part is difficult to manage commercially and tends to be time and material. It is less of an issue than the other elements since it has no role after the project and the proportion of cost in this role is quite low, typically less than 5%.

The second role is to secure implementation services in addition to or instead of the software vendor. In the scenario with one software vendor delivering the majority of functionality, this is typically a time and material item and the system integrator does not have full responsibility for the solution. The second role can also be filled with more regular staff augmentation.

Some parts of the services, e.g., in testing, can be in any of the roles. The two roles should preferably be filled by different companies for both commercial and competence reasons, but if there is competent internal oversight, this is not strictly necessary.

The **second scenario** is where the system integrator has responsibility for the entire solution. In this case, the first role of control should not be combined with the second role of implementation. Here, the tender would go towards a system integrator who would partner with a software vendor to deliver the solution. In case of multiple software vendors, e.g., operational CRM and billing separate, the role of the system integrator becomes very prominent.

The reason for separating the roles in this case is that the system integrator will have a commercial interest in moving away from the standard system since it will increase the work outside and therefore the revenue of the system integrator. Most system integrators with substantial implementation practices have sales incentives that will emphasize this.

A key disadvantage of having a system integrator in a very prominent role is that, for telecommunications, the system integrators rarely have sufficient insight in the software to deliver it in a good way. This means that the ability to optimize overall value creation may be severely compromised. This is contrary to most ERP solutions (e.g., SAP, D365) where the delivery model is based on system integrators who have staff with deep system insights.

5.7 Use of standard systems

The promise of standard systems is that they provide a rich functionality with high quality at a fraction of the cost that a similar custom system would require. Furthermore, they continuously come in new versions that give new functionality in line with the "best practices" of the business. Also, they facilitate simplification through standardisation. And finally, they provide cost predictability.

There are some downsides and caveats also. The most important downside is that it can be a very challenging task to understand the application and utilize it in the best way. Understanding the application is a requirement for being able to operate it (and failure to secure this capability will over time remove commercial leverage for the customer, see section 5.2 for further discussion of this topic). Further, in a vendor-driven project with focus on deliverables, nuances of requirements may be missed causing a necessary re-implementation after the project when the system capabilities and limitations have been fully understood.

Handling the issue of understanding the application is discussed further under the discussion of knowledge transfer, section 6.6. The questions of what constitutes a standard system and if standard systems are always viable are discussed below.

5.7.1 Definition of a standard system

The issue of what constitutes a standard system is the focus of this subsection. The core of the problem is the question of ability to upgrade. When a system is non-standard, upgrading it to new versions or even handling new database or security standards, can be a major undertaking. Upgrading is simpler if one has a standard system where functionality is implemented with "pure configuration".

So, what constitutes "pure configuration"? Ten-twenty years ago, that was a fairly simple concept. Configuration was a set of fixed entries in a table or a file that defined the behaviour of a system, e.g., currency conversion rates or duration before starting a dunning flow. Such configuration items still exist, but for some systems scripting a rule engine or even software code can survive an upgrade without any intervention; for SaaS applications such an ability is frequently a requirement.

Since the benefits of standard system, apart from its functional abilities, is closely linked to the ability to do seamless upgrades, the concept of a standard system becomes a legal rather than a technical

item. A standard system has the following characteristics:

1. The functionality it provides is well documented.
2. It has regular updates in the form of releases with release notes, explaining what the upgrade is and how it is applied.
3. Applying a release takes place using automated scripts and a reasonable release test. Not an upgrade project.
4. It has firm rules for what is *permissible customizations*, i.e., changes or configurations that may be applied without losing the advantage of being able to upgrade using scripts delivered by the vendor and basic verification only.
5. The system is constructed in such a manner that the advantages of the system may be obtained in real-life operations within the limits of permissible customizations.

In order to qualify as a standard system, the vendor should be willing to put this in a contract, including taking responsibility (and liability) for upholding the result. This way the problem of defining what 'standard' means in technical terms goes away and is transferred to a legal requirement.

Note that a database system is a standard system in this definition. Which is fine, but fairly uninteresting for the purpose of implementing business support systems.

Securing long-term cost control requires that the core system is and remains a standard system. In case each upgrade requires vendor participation, the cost of such participation is largely at the discretion of the vendor.

5.7.2 Suitability of standard systems

The point of view taken here is that for telecom operators, standard systems are suitable for all the core transactional processes. That does not imply that their footprint should be universal. There can certainly be areas where customized solutions can make sense. One obvious candidate is online where frequent changes and tests of customer behaviour requires fast changes and full flexibility. Another area is the still-explorative area of machine learning.

Again, in the view taken here, there are two key requirements for using custom systems:

1. It is a conscious decision based on real business benefits derived from flexibility and control of the system.
2. It is clearly contained architecturally (so that for instance the online solution does not start making its own price calculations independently of the main product catalogue).

5.8 Identifying vendors

Prior to executing the contracting phase, the list of vendors to invite must be produced. Several sources exist for this, with the analysis firms like Gartner and Forrester being the most prominent. For the very initial round, it is better to spread the net rather widely. Through setting requirements on the process fairly strict, including that only vendors with complete functional footprint and relevant references are considered, a level of self-selection can be achieved.

The approach outlined here, in particular the contracting approach, will typically discourage vendors who do not wish to accept the terms that secure cost predictability, typically vendors that perceive themselves as having a product and market position enabling them to dicate terms.

5.9 Methodology

In defining the methodology, the practices of industry should be applied, as well as avoiding the pitfalls outlined above. In addition to these, the design principles applied here are:

1. Optimize overall value creation through focus on "what" and utilize vendor expertise to define "how", jointly defining a good fit to the standard product.
2. Maintain commercial leverage throughout and after the implementation.

The real key point here is to optimize value creation. If one insists on specific solutions, processes etc., it will not necessarily fit into the systems proposed. The vendors will know their systems so well that they can propose the best way of achieving the business targets – if one lets them have the freedom to do so. This is one of the places where the use of system integrators replacing the software can be challenging.

The suggested approach is outlined below, showing the process once the gross list of vendors has been found.

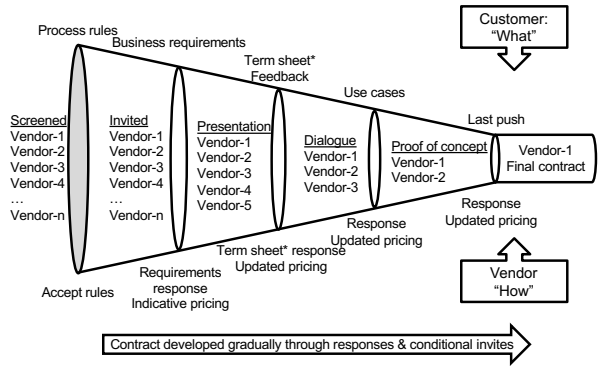

Figure 4: procurement process

The process looks like a classical procurement funnel: many vendors enter the funnel, and one emerges as the selected. It also resembles it a lot, but it also has important differences; it is designed to develop the solution jointly with the vendor.

In the illustration, the customer actions are at the top and the vendor responses at the bottom. So, for instance at the start, the customer publishes the "process rules" and the vendors "accept rules" (assuming, of course, that they actually do accept the rules).

The funnel is shown as a strict waterfall with one ring following the next. When converting the method to a plan, this is not the reality. Developing the business requirements, the term sheet (or contract, see section 5.9.3 for a discussion of the contracting approach) and the use cases are time-consuming activities and should be started in parallel.

At each ring of the funnel, the vendors can be evaluated and sorted. The process is flexible with respect to the number of participants except for the last phase where the numbers should be down to two.

5.9.1 Process rules

The first step is to invite vendors including setting out a set of process rules. The process rules should explain how the process works, including:

1. Description of the process, timeline, contacts.
2. Rules governing the communication, e.g., who it is permitted to speak with, process for questions etc.
3. Rules governing the negotiation process, e.g., whether 'second round' bidding is permitted.
4. Any response during the process shall become part of the final agreement. This includes, in particular, that minutes and recording of all presentations commit the vendors and become part of the final agreement.

Here, the last rule is very important as it requires the vendors to participate in the gradual development of the solution in a committed manner.

5.9.2 Business requirements

The second step is to develop and publish the business requirements. The business requirements describe the business to be supported, including products, processes, number of customers per segment, compliance, security, technical environment into which the solution must fit etc. This is typically a document of 50-200 pages and is core to the process as it represents the foundation of what needs to be delivered. It, therefore, typically needs approval with a broad number of stakeholders. The document should be developed by a small core team to secure consistency and concurrently validated with relevant stakeholders. The vendors respond to the document with a statement of compliance and a non-binding price indication.

The greenfield approach to some extent builds a "new company" on the side of the old one and moves the customers to that. This, of course, is an exaggeration since the basic brand, infrastructure and all processes that are not the customer facing transaction processes are unchanged. Nonetheless, it does represent a very significant renewal of the business.

Writing the business requirements assumes that a view of future business exists or can be created in the process. As such, a simplification is to some extent a pre-requisite for starting work on the business requirements.

This does not mean that every detail of the simplification is developed prior to the implementation phase. Part of the process outlined is to develop the more detailed requirements jointly with the vendor. This means that one needs to know where the business is going – the "what" part in Figure 4. Segments, brands, products, volumes, territories, governance, overall application architecture – these are things that need to be understood when drafting the business requirements. But other parts of the simplification are provided through the process of joint specification in adherence to a standard system.

When defining new or updated processes, understanding the situation from the customer perspective, often termed the "customer journey" is crucial. This implies keeping the customer perspective, testing against real customers etc. Also important is to maintain a process close to reality; not infrequently, customer journey visions are not viable in practical operations. The customer journey definition should be developed hand in hand with the process definitions – they really are two sides of the same coin. Customer journey perspective helps keeping customer experience in focus. Process definitions help keeping practical operation as the necessary basis. This relationship is

particularly important when employing consultants that only focuses on one part – in itself that may be fine as not all have both competences, but the development should still be closely co-ordinated.

5.9.3 Contracting basis

The third step is contracting basis. There are a couple of options for this.

Obviously, one can merely accept the vendor's standard contract and negotiate from there in the final step. That, generally, is commercially challenging once the vendor is alone in the process.

To avoid this, two different approaches can be applied: writing and submitting a full contract for the vendors to consider or asking adherence to a term sheet.

Writing and submitting a full contract is the most thorough approach and secures that all pertinent aspects are covered before final down-selection. The main disadvantage is the effort required in writing the contract, in particular for companies where this is not a regular occurrence and negotiating several different mark-ups. Contracting this way should, therefore, be postponed until the number of vendors are low, preferably only two. In order to secure sufficient commercial leverage, it should be combined with the term sheet approach earlier in the procurement process, where some of the key items that normally are contended should be clarified early.

A lighter approach is to submit a term sheet that covers the items that are normally contended in software contract and ask for compliance. This will then be incorporated in the standard contract of the vendor. Before submitting the term sheet, the vendor should also be asked to provide the standard contract so additional terms may be added. The advantage of the process is that it requires a lot less resource. The key disadvantage is that it leaves certain negotiations until the final contracting, where the commercial leverage is poorer.

Jointly with the contracting input, the vendors are given feedback on their response, both the content and the price. The vendors then respond to the contracting basis and reverts with updated pricing as well as updates to the response to the business requirements, if applicable.

5.9.4 Use cases

The fourth step is issuance of use cases the vendors must support. The purpose is to ensure a structured walkthrough of key functionality that may be documented as a contractual commitment.

Therefore, any functional description technique may be applied – use cases is one option that has the advantage of being fairly easy and efficient to use.

The functional description need not be fully covering all requirements but should cover the core functionality. The vendors shall respond with compliance to the use cases, preferably including a description of how the use cases are supported, and potentially updated pricing. Also, the use cases form the basis for the proof of concept.

5.9.5 Proof of concept

Before entering the fifth step, the number of vendors should be down to two. In this step, the vendors demonstrate how their solution will support the business requirements and the use cases. This should be done through a "proof of concept workshop" of 4-8 days where the relevant use cases structure the walkthrough and the compliance is recorded, electronically as well as in minutes.

The best result is achieved when the vendors are paid for this activity. This goes in particular for smaller contracts where the vendor sales management may be reluctant to spend substantial time of configuration without a contract. This approach is akin to the paid "architects' competition" employed in some tenders for construction.

5.9.6 Contracting

Finally, a vendor is selected, and the concluding negotiations can take place.

6 Greenfield phase 2: implementation

The implementation phase can obviously be executed in a number of different ways with differing approaches for development, testing, project management etc. The approach described here continues from the results coming out of phase 1 and has been tried in practice in various flavours.

The section starts with a methodology discussion followed by a review of a sample project organisation. Not that the organisation structure is really that important, but jointly with the methodology, it serves as a framework for discussing the various roles and responsibilities.

Following this, a number of key implementation considerations are discussed:

1. People and change.
2. Test.
3. Migration.
4. Knowledge transfer.

6.1 Methodology and plan

As will be recalled from the above, the approach addresses a situation where a vendor – either a software or a system integrator – delivers very large proportions of the project. Other deliverables, e.g., interfaces, adaptations to existing systems or new online presence, are delivered in the project or by other vendors, internal or external. All this needs to come together in order to secure the delivery. In addition, in order to keep cost control, the requirements for the vendor needs detailing prior to execution. Therefore, the overall program approach resembles an old-fashioned waterfall.

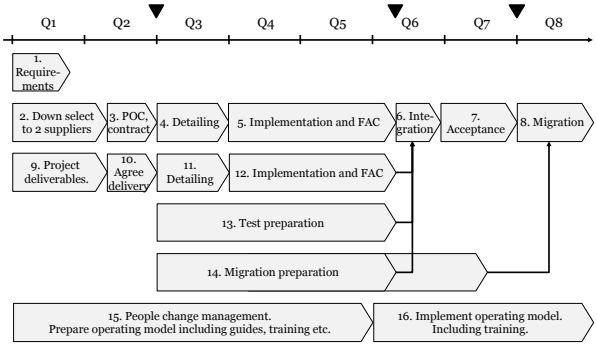

Figure 5: generic implementation plan

The shaping phase are in Q1 and Q2 and the implementation phase from Q3 onwards with testing happening in Q6 and Q7 and migration following that.

Note that the waterfall approach is identified via the key milestones indicated: contracting, start of test and start of migration. Here, all the components need co-ordination in time and content. How each delivery is managed, via agile or waterfall, is not material to the process.

A normal reaction is that this is a long time. Two years, of course, is that. However, the timeline outlined is an optimal one requiring extremely efficient execution. Three years is a more common execution time. It is an almost invariable experience that pushing the timeline too hard will result in delays that makes the project take even longer.

The activities 1-3 are described in the contracting phase, section 4 above.

Activities 9 and 10 identify and secure delivery of such items that the main process (activities 1-8) are dependent upon, e.g., preparation of dual operation, data cleansing. The dependencies can be managed by the project or can be delivered by the organisation or an external (typically not the vendor of the main system). Irrespective of who and how, agreement of delivery must be in place in order to execute at the time indicated.

Activities 4 and 5 will depend on the approach from the vendor. The detailing will normally be required as the specification developed in the contracting phase will not be complete. At the end of activity 5, the vendor should deliver a solution that has been through internal testing.

Activities 11 and 12 correspond to 4 and 5 and focuses on implementation of the dependencies, typically related to temporary interfaces or interfaces to existing systems that remain after the project. Here the approach can vary quite a bit, and for internal deliverables more flexibility on end-result is possible. For instance, the online presence can be more or less sophisticated in an initial release and that can be managed flexibly internally. Similar to the deliveries from the vendor out of activity 4 and 5, the deliveries must be tested and functioning internally.

Activity 13 is preparation of test which is further described in section 6.4. It prepares test of integration, acceptance and, if applicable, manual processes. Note that the start of the activity is early – preparing a proper test model is extensive work requiring on-going interaction with the detailing and implementation.

Activity 14 is preparation of migration. Like test, it starts early as there is substantial work in designing dual operations, identifying data sources and initiating data cleansing.

Activity 15 and 16 are the preparation and implementation of the required change. See further discussion in section 6.3.

Activity 6 is the integration where the various elements meet “officially”. Clearly, it is advantageous if pre-integration has taken place. The integration testing is a key milestone, the latest time at which all systems meet and a precondition for meaningfully entering acceptance.

Activity 7 is the acceptance where the prepared test scripts are executed in order to verify readiness for go-live. For external s, in particular the vendor of the main system, it also has a contractual role.

Activity 8 is the physical migration. See further in section 6.5.

6.2 Project organisation

In the contracting phase described in section 4, the project organisation is small and with little or no formal structure. The process is explorative and knowledge sharing is key.

Moving to implementation, more structure is required. In the implementation phase up until start of integration, a typical organisation structure is as follows.

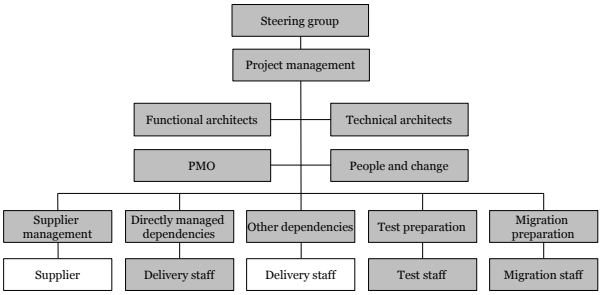

Figure 6: generic implementation organisation

The structure is typically adjusted in the test phase to accommodate the focus on error correction. In case of multiple releases, corresponding adjustments are required. The white boxes are dependencies that are not managed by the project. In principle, test and migration could follow the structure of 'other dependencies' but since this work starts and ends with the project, the normal approach is to include it with the project.

The project management is a combined content and progress role, with primary responsibility of securing overall business goals.

The functional architects are key people designing the future business. They are responsible for formulating the business requirements and the further detailing. They should be mandated to define the future business – i.e., to take all the detailed decisions that are required in order to design a new IT system, even when it is based on a standard system. Through a reference group or other structures, suitable involvement and approvals in the line organisation can take place.

The functional architect is often the most difficult role to fill since it requires knowledge of the current business, a vision of the future and sufficient seniority to balance the vision with realities of project execution. Such individuals are invariably already in key roles; if it does not hurt severely when taking them out of the line organisation, they are likely to be the wrong people.

The technical architects are responsible for securing alignment with technical standards, interfaces etc. They write the technical parts of the business requirements and detail the interfaces, data mastering and other similar items. If preparation of operations is included, infrastructure architects design and secure this.

Typically, a fairly large number of complex issues needs to be resolved, requiring both function, business understanding, insight in current systems and ability to analyse and facilitate agreement on technical solutions. While typically simpler to staff than functional architects, these are also highly specialized roles.

The PMO is the administrative part of project management, handling resource allocation, financial follow-up, physical accommodation etc. This is more generic and can be staffed from the system integrator or as freelancers.

The people and change are responsible for driving the change within the line organisation. Again, this is fairly generic, but preferably the change competence should be combined with people understanding the organisation well.

Vendor management secures link to the vendor and their deliverables, managing the contract, resolving issues etc.

Directly managed dependencies is a subproject that handles the technical deliverables that are directly from the project, i.e., handled with staff allocated to the project.

Other dependencies task is a subproject that handles deliverables from other parts of the organisation, external vendors etc.

The three last roles, the vendor management, the directly managed dependencies and other dependencies, require project management and business understanding, interacting with the functional and technical architects. They can be structured in different ways. Here it is illustrated based on relationship with the programs, but it could also be a functional delineation, e.g., one part responsible for the channel part of the systems or value stream based.

Test preparation is the team that prepares the test model. In some cases, the software vendor does this, and then some of the staff would not be directly managed.

Migration preparation is the team that prepares for data migration, data cleansing and other similar activities. Securing dual operation can also be placed here.

6.3 People and change

In section 2, the main organisational change challenge of mobilizing the entire organisation towards the transformation goals was discussed.

At a more pedestrian level, people and change is concerned with preparing the organisation for the change from the new tools. They may imply changing roles, e.g., higher first-time resolution will shift staff from back-office to front-office. And they will certainly require updates guides to front-end staff, training etc.

Securing that these activities all take place is the role of people and change. The experience from the projects that form the basis for this document is that the project itself should be focused on orchestrating the change. The execution of the change activities should take place in the line organisation. This includes for instance:

1. Preparing and implementing change to the organisation structure, if applicable.
2. Developing process and instruction material for use of the new systems.
3. Preparing and executing training.
4. Updating KPIs, department and individual targets etc.

6.4 Test

Test happens at many levels in major system replacement projects. Each of the deliverables that jointly constitute the solution have internal tests, interface tests, process tests etc.

The topic of this section is the process test that verifies that all the functionality is in place. A very light version of it constitutes the integration test that precedes the process test.

Once the process test has been executed successfully, the solution is ready to go live and be accepted.

Testing this way resembles classical system test in a waterfall project. However, the effort is reduced from two factors. Firstly, large parts of the solution are based on standard systems, so a lot of basic errors will be eliminated even before start of the project. Secondly, the migration approach is assumed to be gradual, which means that the tolerance for errors is substantially higher than in big bang migrations. This risk balance is an important determinator for the complexity and duration of the test.

In order to execute the test, a test model must be in place. A test model consists of a number of test cases, typically 1,000 – 2,000, that are joined into scripts for execution. Part of the test model is kept long-term for regression test. Once the test model and the system are stabilized, it may be considered automating the test execution.

The test model typically consists of a large number of functional test cases and fewer, but crucial, security and compliance test cases, including business continuity etc. These are registered in a collaboration tool that permits registration of execution, errors, re-testing etc.

In order to execute the test, there needs to be comprehensive environments spanning all relevant systems. A number of such environments are typically required in order to execute integration test, acceptance test, technical tests, training etc. Depending upon the maturity and sourcing model of IT, this can be anywhere from straightforward to extremely challenging.

Furthermore, data is required. In these days of GDPR, data can either be constructed or anonymized. Both options can entail significant effort.

Since the software vendor in many cases builds substantial parts of the test model, it is tempting to use this for acceptance test. That is also quite viable, but firstly the vendor normally only covers part of the scope and secondly the acceptance test also has a commercial implication in signifying the completion of the vendor's commitments in the project. As long as these issues are addressed, it can save substantial amount of work.

6.5 Migration

Migration of customers to a new system can take place gradually or over a couple of days, the latter termed 'big bang'. The approach outlined in this document assumes gradual migration. The key reason is the risk associated with big bang migrations, where faults can cause customer operation to halt, ultimately threatening existence of the company. In order to avoid such risks, a big bang approach requires a more extensive test than what is indicated in the plans presented above.

The gradual migration assumes dual operations, i.e., the parallel execution of the legacy and the new system for a period of months. During this timeline, a number of areas need to interface, including online, call centre PBX, payment interfaces, CDRs, number porting etc. In addition, the resources like phone numbers, SIM cards, CPEs, IP addresses etc. need to be managed jointly. Some resources can be pre-allocated to either stack, but others like physical addresses and access ports must be shared.

The dual operation needs to be prepared and preferably put into production well before the go-live of the solution (to avoid confusing where the errors are coming from).

The migration can be driven by the customers, in which case dual operation is a year-long affair, or through moving customers actively from one system to the other. In the latter case, the tempo needs to be adjustable in order to mitigate the risk; this means that changes made to the customer cannot imply that the customer migration date becomes fixed (like it might be if the customer should have notice of the change).

Technical migration can be product based or customer based. In the product-based approach, all

customers with product X are moved to product Y. This is analytically simple and often used. In case of many high-ARPU customers moved to lower-ARPU products it can also be expensive. Another option is to make it customer-based, defining the new product based on customer profile and usage. This is more complex but can in some cases mitigate the ARPU decline, since the customers can be migrated to "value loaded" products, i.e., getting a product with more consumption or features rather than a decrease in price. The customer-based migration may require customer notification, which then shifts the risk balance of the migration.

Migration is nearly always challenging. Data is typically inconsistent and/or incorrect, data sources are hard to identify, and dual operation adds to complexity.

6.6 Knowledge transfer

As was noted in section 5.2, getting the right target operating model is important to secure continued commercial leverage. In order to enable a target operating model where the vendor of the core software does not continue to deliver services except for support and maintenance, other staff, internal or external, needs competence.

The competence includes application operation, application maintenance and application development. The latter two in particular include ability to configure the system as well as interfaces.

Due to the complexity of the systems contemplated here, such competence cannot be achieved fully through training. People need to work with the systems to understand their potential and shortfalls, selecting and advising on the best way to use the system. This is best accomplished through having people participating in the development jointly with the vendor. If this is not possible, it must be expected that the vendor needs to participate in all activities for a while following go-live. Firstly, this should be budgeted and secondly, participation of internal staff in the activities following go-live should be enforced.

7 Summary

The greenfield approach outlined here provides a process for implementing a new core system. The first phase, contracting, is the most important since it secures the simplification and adherence to standard systems. In particular, for the simplification, strict adherence to standard systems is important.

The second phase is more regular in the sense that it resembles other approaches to system implementation.

While not applicable to all contexts and still requiring substantial experience to execute, it contains a framework that make such projects relatively predictable.

As stated initially, it is a key purpose of the process to ensure long-term cost predictability. This is secured through the following measures:

1. Strict adherence to standard systems.
2. Contracting for fixed-cost implementation.
3. Securing ongoing commercial leverage in all areas except support and maintenance.
4. Contracting for cost predictability in the support and maintenance.
5. Securing termination notices that does not undermine cost predictability for support and maintenance.

This, of course, provides a *basis* for cost predictability. If the subsequent governance permit deviation from the standard system, it will be a question of time before one return to the "legacy" problem. While the people and change part of the project can focus on this, it remains a management task and challenge to keep adhering to simplicity and standard systems.

8 Contact

The author of this document is Lars R. Andersen. Lars works as an advisor, primarily within networks and IT and has a background from consulting, primarily from Accenture, and CTO/CIO of Telenor in Denmark and can be contacted via: lars@ra-advisory.dk. This and other white papers are available free on www.ra-advisory.dk.

Obviously, there is a lot more to BSS replacement than what can be contained in this document. If you wish further perspectives, access to templates etc., please feel free to reach out.