cutoverconcepts

How to implement an ERP.

Version 0.18

Contribute your lessons at https://cutoverconcepts.org/

Contents

Introduction

What is Cutover Concepts?

Cutover Concepts (CC) is an open-source framework for getting your organisation live on any of the big ERPs. CC is built from shared lessons and is described in this guide.

Why does this guide exist?

Most ERP implementation programmes encounter common pitfalls. This guide explains how to avoid them and hit your go-live date.

How to use this guide

- ✓ Read this handbook for perspective
- ✓ Check suggested roles and responsibilities
- ✓ Get the programme started
- ✓ Implement your new ERP
- ✓ Share your lessons to shape Cutover Concepts

What is an ERP?

Enterprise Resource Planning (ERP) solutions are integrated systems combined with people and processes to carry out essential activities, e.g. paying your staff.

What is a cutover?

Moving from an existing system to a new ERP without is known as a cutover. It takes a methodical approach to do this reliably without interrupting business activities, and getting ready for it can be tricky.

Some useful terminology

'Solution': The ERP technology, supporting systems, people and processes.

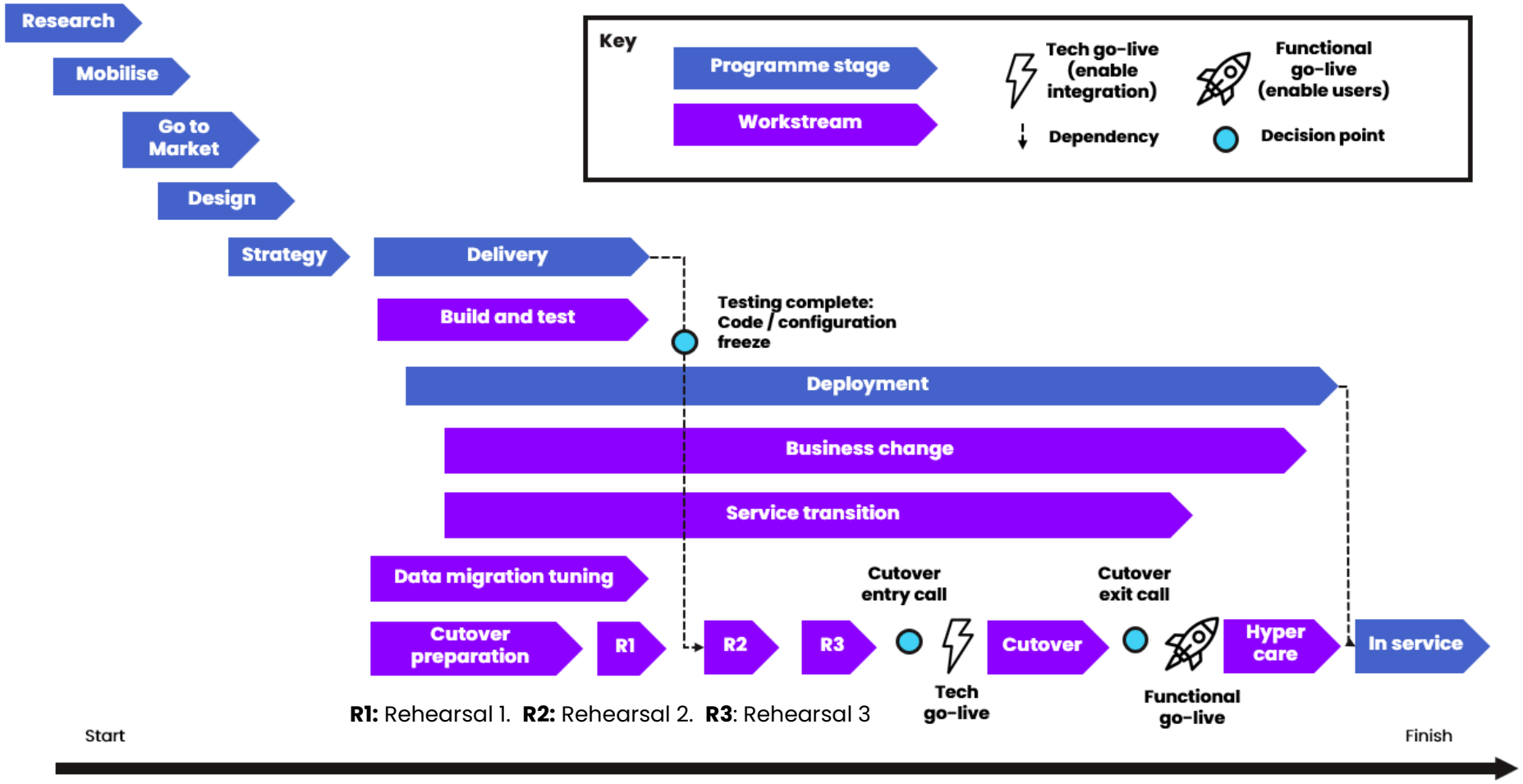
'Delivery': The stage covering build and test of the ERP system.

'Deployment': The tasks needed to get the overall solution live.

'Cutover': The event of moving an organisation from one live solution to another (with 'cut-over', often used as the verb).

Framework overview

Typical high-level phases and tasks.



R1: Rehearsal 1. **R2:** Rehearsal 2. **R3**: Rehearsal 3

Stage 1: Research

First, understand why you need a new ERP and your company's current situation:

1) Decide on the desired outcomes and write them down.
2) Document your ***current*** business processes, skills and systems.
3) Define your available budget and time constraints.
4) Research available ERP technologies and what they can do.

Pitfall: Current processes or systems are not documented.

Without this information, transitioning from the current to the new is more difficult to plan.

Solution: *Begin mapping processes and systems if this hasn't been done.*

Stage 2: Mobilise

On board a team of experienced and independent team members to sense-check documents and decisions. The people you will need to start with are:

1) Business Lead (representing the customer)
2) Programme Manager
3) Project Manager(s)
4) Programme Architect
5) Business Analyst(s)
6) Implementation Lead

Here are some basic programme artefacts they'll need to create and/or manage:

- ✓ Outcomes (now further developed)
- ✓ Programme milestones
- ✓ Board reports
- ✓ Agreed budget
- ✓ Contracts
- ✓ Project plan(s)
- ✓ Risk log
- ✓ Resource plan
- ✓ Stakeholder map

Stage 3: Go to market

Your organisation will probably adopt business processes that fit the ERP. This is to minimise customisation effort and maximise uptake of new features. In addition to getting the technology live, you may need help to transform your organisation.

Going to market can look like:

1) Find a suitable procurement framework or marketplace.
2) Express your business taxonomy, goals and current systems.
3) Explore multiple ERP technologies.
4) Include multiple Implementation Partners and System Integrators.
5) Ask the consultancies to recommend new tech and benefits.
6) Request strategies from the SI to describe their approach.
7) Use your independent team to sense-check proposals.
8) Pick a technology and implementation partner (or SI).

Useful terminology

'Implementation Partner' A consultancy that offers business and technical services for an ERP programme, including business transformation.

'System Integrator' A consultancy focusing on building the new solution, but not necessarily business change.

Pitfall: Accepting a plan that contains only one rehearsal (or none)

If there is only one rehearsal and it fails, there won't be confidence to go live. So, there might be a costly re-plan.

Solution: *Commit to more than one rehearsal.*

Stage 4: Design

Based on the initial research and SI proposal, really get to grips with the design. Things to cover in this stage:

- ✓ Design Authority setup
- ✓ Technical vision
- ✓ High Level Design (HLD) then Low Level Design (LLD)
- ✓ Legacy and reach-back design
- ✓ Data flows
- ✓ Data model
- ✓ Functionality
- ✓ Needs, journeys
- ✓ Requirements, NFRs (Non-Functional Requirements)
- ✓ Business architecture
- ✓ Role mapping
- ✓ Integration specifications
- ✓ Environment model
- ✓ Data owner table (legally accountable data owners)

Pitfall: Starting build of the test system without an approved design.

Minor tweaks after starting the build can usually be accommodated. However, changes in the HLD or in essential operations can result in costly rework or delays.

Solution: *Agree on the design before starting to build. Get the design signed off by technology and business leads.*

Stage 5: Strategise

Figure out how to build and deploy your new ERP solution. These strategy documents influence each other and pave the way for viable plans:

Test strategy: How to demonstrate the technology works for the organisation.

Tech vendor methodology: Constrains mandated by the Tech vendor to go-live on their platform or software.

Deployment strategy: How to prepare the people, processes, data and support teams ready for cutover and beyond.

Cutover strategy: How to move from the old system to the new. This is heavily dependent on the deployment strategy and data migration:

Data migration strategy: This describes how to prepare, extract (from the source system), transform, load (to the target system) and validate the data.

Environment strategy: How to provide and manage environments to enable the other strategies and frameworks.

Pitfall: Skipping strategy creation and going straight to plans.

An ERP implementation has many moving parts. If the interactions between workstream outputs are overlooked, data migration tasks can be missed, testing can be insufficient, and last-minute resourcing gaps can appear.

Solution: *Early on, establish strategies and agree on them across the programme. This will highlight gaps to fill and inform joined-up planning.*

Stage 6: Data migration

Moving sufficient and accurate data from 'source to target' systems is known as Data Migration (DM). This will need to cover static and dynamic, including in-flight transactions. Also, feeding downstream systems and archiving should be considered.

Key points:

1. Data migration must be optimised to ensure it can be done within cutover time constraints.
2. Data must be validated by Functional Leads at its destination to make sure it's correct.

Information needed in data migration:

- ✓ ETL (Extract Transform Load) logic
- ✓ Data source list
- ✓ Data mapping document
- ✓ Data reconciliation reports
- ✓ Data cut-offs (last entries to source system)
- ✓ Data protection documentation
- ✓ ETL duration time constraints

Pitfall: Taking a low-cost proposal without delta migrations.

'No-delta' approaches often rely on the manual capture and load of data off-system, or even dual keying. This can be a huge undertaking.

Solution: *In the strategise stage, understand if the data migration strategy would work for cutover. Consider delta or 'top-up' migrations to minimise any data freeze periods.*

Stage 7: Deliver

Build and test the system. This should happen alongside early deployment as there are several **interdependencies***. Key information you will need:

- ✓ Environments plan, specifications, build plans
- ✓ Technical skills matrix
- ✓ Delivery plan
- ✓ Workflow management setup
- ✓ Test plans
- ✓ Environment booking table
- ✓ Release process
- ✓ Non-production access control
- ✓ Code / config management process
- ✓ Code documentation / config workbooks
- ✓ Code / config
- ✓ **Live proving scenarios***
- ✓ **Defect and workaround tracker***
- ✓ Testing sign-off
- ✓ End of Test Report

Pitfall: Waiting until the end of testing to setup workarounds

Complex manual workarounds may require a long setup time.

Solution: *Watch for critical and major defects in the weeks leading up to the end of testing. Start to get workarounds ready well before testing ends.*

Pitfall: Looking at workarounds in isolation.

This will mask where the same team is needed to execute multiple workarounds. They may not have the capacity for all of them.

Solution: *Call out the resources needed and examine the combined list of workarounds and the aggregated effort.*

Stage 8: Deploy

This stage crosscuts all areas of an ERP programme. This is how each workstream can feed into the deployment stage to prepare for cutover.

Business change

This workstream covers training, communications, business processes ***and role mapping***. Key information to be produced:

- ✓ User list
- ✓ Role mapping (which job types get which system role/permissions)
- ✓ Training material
- ✓ Business process changes
- ✓ Ways of working changes
- ✓ Business readiness criteria
- ✓ Communications plan

Service transition

The Service Transition Lead acts as the 'gatekeeper of production'. They set transition criteria to be met before go-live. This presents a healthy challenge to the programme's 'drive to go-live'. Key items are:

- ✓ Service transition criteria
- ✓ Service transition plan
- ✓ Hypercare support model
- ✓ Hypercare entry / exit criteria
- ✓ Support model
- ✓ Knowledge transfer
- ✓ Knowledge articles
- ✓ ITSM (IT Service Management) tool setup
- ✓ Production change order
- ✓ Support team impact analysis
- ✓ Support team skills matrix
- ✓ Up-skilling and resourcing plan
- ✓ Monitoring and alerting setup
- ✓ Disaster recovery arrangements
- ✓ Update and evergreening model
- ✓ Daily / twice daily incident call
- ✓ Change approval board setup
- ✓ First occurrences to be monitored
- ✓ Usage telemetry
- ✓ Hypercare exit criteria

Cutover preparation

There should already be a cutover strategy in place. Develop this into a high-level cutover plan, including business, technical and **full scope production data migration** activities.

These documents are needed in preparation for cutover:

- ✓ Deployment plan
- ✓ Readiness criteria
- ✓ Readiness report
- ✓ Rehearsal scope
- ✓ Entry and exit criteria
- ✓ Cutover POAP (Plan on a Page)
- ✓ Runbook template
- ✓ Runbooks
- ✓ Detailed cutover plan (ready for rehearsal)
- ✓ Backout plan (to get services back online if the cutover fails)
- ✓ Cutover sitrep (situation reports)

Runbooks

Use the high-level plan to lead a 'familiarisation campaign' to:

- Get people on board
- Recommend a cutover format
- Introduce runbook templates
- Begin population of the runbooks

Walkthrough

Runbooks are combined to create the detailed cutover plan. This is then checked in a 'walkthrough' where all team members talk through the tasks start to finish.

Rehearsals

The goal of rehearsals is to demonstrate that the team can successfully execute the cutover as planned, given the time and resources available. They also identify unknown factors that would scupper a cutover, e.g., out-of-hours resource power-downs.

Rehearsal 1 (R1)

R1 is technology-focused, to prove:

- The main components of the system can be connected
- Data sets migrated (not necessarily full volume)
- Key user group representatives given access

There will be many challenges and lessons to be learnt in R1, so set aside plenty of time. Lessons from R1 should be

Rehearsal 2 (R2)

This is the full and should include:

- The full, actual team that will execute the cutover.
- Full volume data migration.
- Timings as they would be on the actual cutover.
- Entry and exit calls should be included, to get Heads-of comfortable with decision making and the information they will be presented.

The full rehearsal is done to prove the cutover can be done logistically and identify any unknown factors that could derail the event. The list is long and unknown, until you try it.

Rehearsal (R3)

It's prudent to plan-in an additional contingency rehearsal, known as R3.

Cutover entry call

This call is to get the Heads of each function and technology capability together and confirm **a)** all readiness criteria have completed or mitigated and **b)** you can enter cutover. Such criteria are:

Pitfall: Expecting to resolve un-met criteria in the entry call.

Its risky to agree entering cutover without green lights across the board. Senior leaders may need time to consider the risks and mitigations, plus get advice from their subject matter experts.

Solution: *Hold readiness checkpoints in the run-up to the cutover entry call, to drive inspection and resolution of incomplete tasks.*

Cutover execution

Everything that has been rehearsed now happens for real. This will vary in each cutover. The main steps are:

- Warm up comms to users
- Enable cutover team access
- Tech go-live (enable integrations and processing in the new ERP)
- Hypercare starts to manage the new system
- Connectivity checks
- Migrate ***bulk*** real data to the new ERP and check it
- Confirmation comms to users: last entries in the old ERP
- Source systems frozen
- Process in-flight transaction data
- Migrate the ***delta*** data to the new ERP and check it
- Check functionality that can be reversed if needed
- Cutover exit call – get a fix-forward or backout decision
- Point of no return
- Functional go-live (enable a small set of business users)
- Live proving (carry out a set of critical operations)
- Enable all remaining users

Useful terminology

'Fix-forward only' The new ERP becomes the system authority and single source of truth for data. If there are any issues, they need to be fixed rather than returning to the previous system.

'Functional go-live' This is the exciting part. All remaining users are enabled, and communications are sent to announce the go-live. The new solution is now operational and the source of truth for data.

'Point of no return' The latest point at which the backout plan can be comfortably initiated to return to the previous system, without impacting business operations.

Hypercare

In hypercare the solution is now processing live transactions. However, the programme team still assists by participating in daily (or more frequent) calls to capture and resolve issues. Key activities and information in hypercare

- ✓ Hypercare incident report
- ✓ Daily / twice daily incident call
- ✓ First occurrence monitoring results
- ✓ Usage telemetry
- ✓ Hypercare exit criteria (results)

Hold a hypercare exit call to make sure the solution works well and the support teams can comfortably look after the new system.

Pitfall: Not rehearsing hypercare

The team may struggle to turn around a fix with fresh, untrodden ways of working.

Solution: *Rehearse assessing and fixing incidents using the hypercare model.*

Stage 9: In service

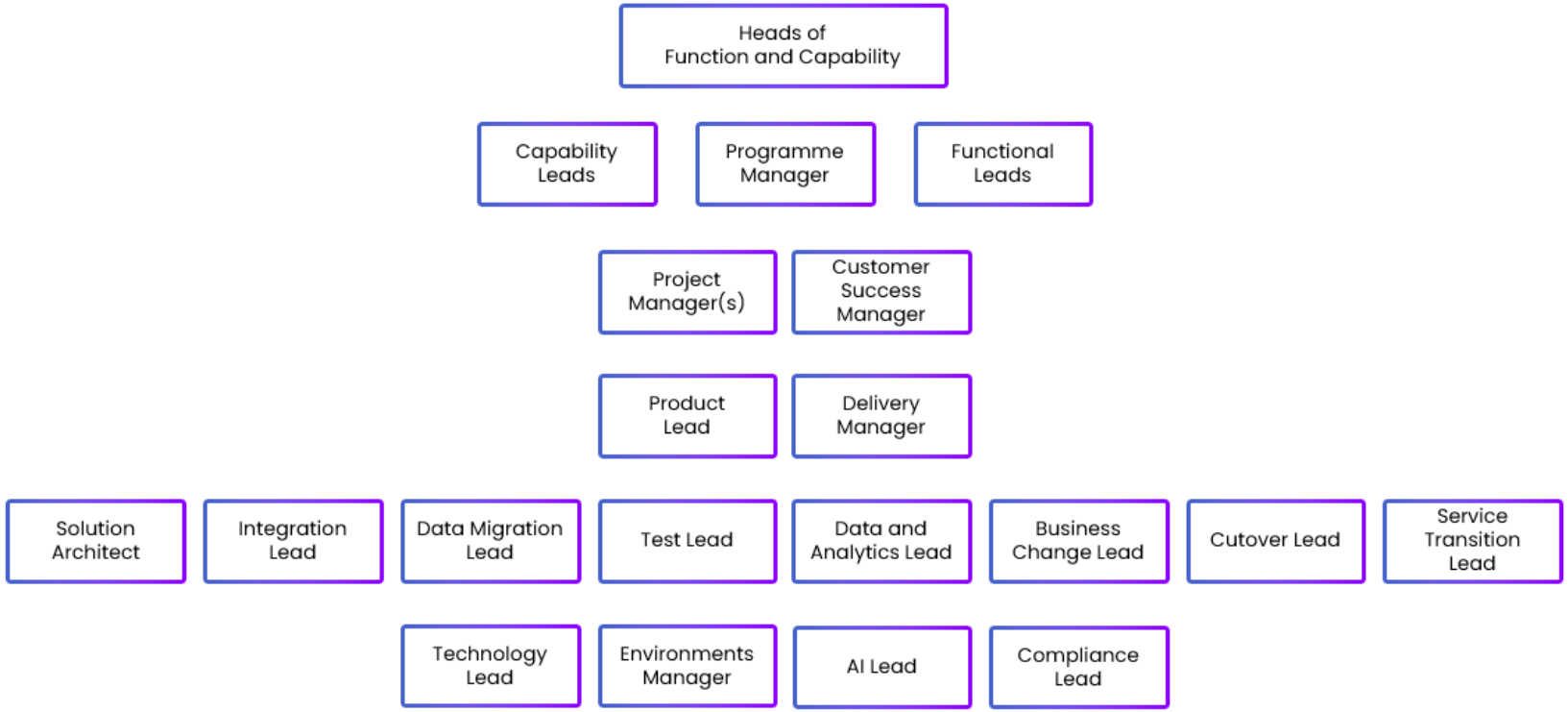
The implementation is now complete. The system, people, and processes (the holistic solution) are now live and supported ***without*** assistance from the programme team.

The solution now has resources and processes so it can:

- Respond to incidents
- Be updated with new features
- Adapt to changes
- Be resilient

The programme team

Team structure



Team composition

A programme team has several leads (or managers), each covering a specific area. Often, a lead has supporting team members. In addition to the core team, technology vendors usually provide a Customer Success Manager to facilitate onboarding SaaS (Software as a Service) products.

Team management

The Programme Manager determines reporting lines, as the make-up of a team can vary. Roles and responsibilities should be agreed upon early on.

Pitfall: Not agreeing on who is responsible for what

Assuming responsibilities purely based on someone's job title isn't ideal. This leads to tasks falling between the gaps and friction where there is 'dual leadership'.

Solution: *Define and agree on (at least) basic responsibilities against an artefact list, to achieve harmony.*

Governance

Ultimately, the programme team reports progress and developments to the Heads-of each Function and Capability. The Heads of' are accountable for the customer organisation. The new solution and how it's to be implemented ***must*** be approved by the Heads of each Function and Capability.

Product centricity and agile ERP

The above structure implies a product-centric team to suggest a feature-led delivery. The Product Lead sits at the helm, managing which features are included in the go-live release, based on the Functional Lead's sign-off.

Pitfall: Assuming agile is the best approach to get an ERP live.

This is a big one. The minimum viable product of an ERP is usually most of its services, as we're dealing with a business-critical system. Also, ERPs should be highly integrated. So, multiple small releases would require very large degree of design, testing and integration effort. Plus, this would risk 'change fatigue' across many users. Some things to consider:

1. *Functional Leads often need to be present out of business hours for each release if critical services are involved.*

2. *Each solution state needs to be designed, approved, built and tested before it can be released.*

3. *Operating multiple General Ledgers, integrating **and** supporting multiple ERP platforms isn't for beginners.*

***Solution:** Be clear on the difference between using agile to build a brand-new service, versus cutting over live services in a set of 2-3 chunky steps. These are known as 'transition states'. Remember, all critical services must be maintained in each state.*

Cutover formats: a selection

Introduction

Here are a few high-level descriptions of different cutover structures, their attributes and basic steps to completion.

Clean cut

This is the simplest format, but it usually results in more downtime to accommodate data migration and a shorter window for live proving.

1. Build the new system and enable the technology without limited user access. Close the current system, make the data static, and then take a backup.
2. Migrate all data from the source to the target system(s).
3. Enable Functional and Capability Lead access and carry out live proving tasks.
4. If all is OK, then give full user access.
5. If not, either fix and mitigate, or backout.

Dual drop

It's the same as a clean-cut cutover, but planning for two releases in close succession. Compared to the clean-cut, this format can help enable a sooner go-live if some 'nice to have' functionality is not yet ready.

1. First: release the essential functionality and data.
2. Second: implement the non-essential remaining features.

Production parallel

This requires early access to the production environment and significant agreements from the Heads of. However, it provides extra time to prove the new system and minimises bulk data migration in the cutover window.

1. Build a parallel production environment and migrate data into it with the agreement of the Heads-of (and ensure legal compliance).
2. Top up the data with deltas (changes) from the operational source system whilst carrying out live proving of the functionality.
3. If live proving is successful, close the source system then migrate the final data delta.
4. Complete the functional go-live.

Transition states

Essentially, this can be viewed as a sequence of individual implementations. It is possibly the highest-effort and cost format. However, if designed and built correctly, it should result in several smaller, potentially more manageable changes to the organisation.

1. Design and agree on a sequence of stable transition states.
2. Ensure each state is functionally and technically viable.
3. Incrementally cut-over to the new solution in groups of functionality or modules.

Acknowledgements

Other methodologies and frameworks of interest

- Scrum.org https://www.scrum.org/
- CDDO https://www.gov.uk/service-manual
- ITIL https://www.axelos.com/certifications/itil-service-management/
- PRINCE2 https://en.wikipedia.org/wiki/PRINCE2

Sources

Written by people, grammar checked with AI.