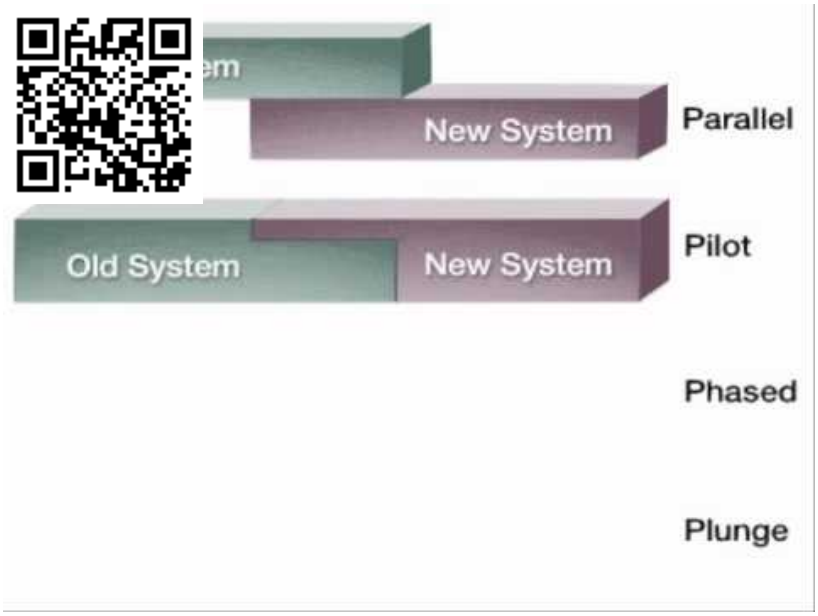
# 10.4: Implementation Methodologies

Once a new system is developed (or purchased), the organization must determine the best method for implementing it. Convincing a group of people to learn and use a new system can be a challenging process. Using the new software and the business processes it gives rise to can have far-reaching effects within the organization.

There are several different methodologies or phases an organization can adopt to implement a new system. Four of the most popular are listed below. Here's a video that gives a brief overview of Implementation methodologies




- **Direct cutover (or plunge)**. In the direct-cutover implementation methodology, the organization selects a particular date that the old system will not be used anymore. On that date, the users begin using the new system, and the old system is unavailable. The advantages of using this methodology are that it is speedy and the least expensive. However, this method is the riskiest as well. If the new system has an operational problem or is not properly prepared, it could prove disastrous for the organization.
- **Pilot implementation**. In this methodology, a subset of the organization (called a pilot group) starts using the new system before the rest of the organization. This has a smaller impact on the company and allows the support team to focus on a smaller group of individuals.
- **Parallel operation**. With the parallel operation, the old and new systems are used simultaneously for a limited period of time. This method is the least risky because the old system is still being used while the new system is essentially being tested. However, this is the most expensive methodology since work is duplicated and support is needed for both systems in full.
- **Phased implementation**. In a phased implementation, different functions of the new application are used as functions from the old system are turned off. This approach allows an organization to move from one system to another slowly.

These implementation methodologies depend on the complexity and importance of the old and new systems.

## 10.4.1: Change Management

As new systems are brought online, and old systems are phased out, it becomes important to manage how change is implemented. Change should never be introduced in a vacuum. The organization should be sure to communicate proposed changes before they happen and plan to minimize the impact of the change that will occur after implementation. Training and incorporating users' feedback are critical to increasing user's acceptance of the new system. Without gaining the user's acceptance, the risk of failure is very high. We discussed in a prior chapter about Change management given that it is a critical component of IT oversight.

**Use case: Change Management 10.4.1**

Acme Inc. recently implemented a new cloud-based enterprise resource planning (ERP) system to manage finances, inventory, and manufacturing. How should they they follow a change management process?

**Solution**

They followed these change management practices:

- Communication - One month before launch, the CIO sent emails and held town halls explaining the new ERP and why it was needed.
- Training - All users went through online and in-person training on the new ERP before and after launch. Quick reference guides were also distributed.
- User involvement - Key user representatives were part of the ERP selection and provided input on configurations.
- Support - For the first month, on-site experts from the vendor helped employees and answered questions. The help desk was also bolstered.
- Feedback - Users were surveyed two weeks after launch about their experiences. Enhancements were developed to address issues.
- Transition time - The old and new ERP systems ran in parallel for two weeks during launch before completely switching over.

By taking these steps, Acme Inc ensured a smooth transition to the new system by increasing user adoption.

## 10.4.2: Maintenance

Once a new system has been introduced, it enters the maintenance phase. In this phase, the system is in production and is being used by the organization. While the system is no longer actively being developed, changes need to be made when bugs are found, or new features are requested. During the maintenance phase, IT management must ensure that the system continues to stay aligned with business priorities, has a clear process to accept requests, problem reports, deploy updates to ensure user's satisfaction with continuous improvements in the product's quality.

Ongoing maintenance is required, including:

- Bug fixes - As bugs are discovered, they must be prioritized and fixed via patches. A structured process for bug reporting, triage, and resolution should be in place.
- Feature requests - Users will request new features over time. Each request should be evaluated and prioritized by business value.
- Testing - Any patches, bug fixes, or new features need to be tested before deployment to production.
- Upgrades - Vendor supplied updates and upgrades to fix vulnerabilities or add new functionality need to be evaluated and rolled out.
- Sunset planning - Eventually, systems need to be retired. Planning ahead for sunsetting legacy systems is key.

To manage these activities, organizations should have defined processes for triaging requests, fixing issues, testing changes, and releasing updates. Regular schedules for upgrades and bug fix rollouts should be established and communicated to users.

With the rise of privacy concerns, many companies now add policies about maintaining their customers' data or data collected during the project. Policies such as when to dispose of, how to dispose of, where to store are just a few examples.

## 10.4.3: Release Management

Once software development is complete, there are ongoing processes for deploying updates:

- Release planning - Evaluates features, bundles them into releases, and sets a rollout schedule.
- Release testing - Testing of new increments before deployment to production environments.
- Rollout - Transferring updates from the test environment and making them live for users.
- Monitoring - Tracking application performance and issues after updates are released.

Organizations need to have a structured release process that ensures changes are properly tested and deployed and to communicate with customers to help with their own planning when to receive and install the company's updates and releases.

---