

Deploying the New System

CSIT883 System Analysis and Project Management



UNIVERSITY
OF WOLLONGONG
AUSTRALIA





Outline

Testing

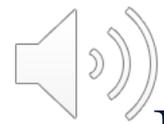
Part I

Deployment Activities

Part II

Managing Implementation and Deployment

Part III



Implementation and Deployment Activities

Implementation activities

- Program the software.
- Unit test the software.
- Identify and build test cases.
- Integrate and test components.

Deployment activities

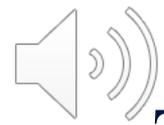
- Perform system and stress tests.
- Perform user acceptance tests.
- Convert existing data.
- Build training materials and conduct training.
- Configure and set up production environment.
- Deploy the solution.

Core processes	Iterations					
	1	2	3	4	5	6
Identify the problem and obtain approval.	[Activity bar]					
Plan and monitor the project.	[Activity bar]					
Discover and understand details.	[Activity bar]					
Design system components.	[Activity bar]					
Build, test, and integrate system components.	[Activity bar]					
Complete system tests and deploy the solution.	[Activity bar]					



Testing Concepts

- **Testing** – the process of examining a component, subsystem, or system to determine its operational characteristics and whether it contains any defects
- **Test case** – a formal description of a starting state, one or more events to which the software must respond, and the expected response or ending state
 - Defined based on well understood functional and non-functional requirements
 - Must test all normal and exception situations
- **Test data** – a set of starting states and events used to test a module, group of modules, or entire system
 - The data that will be used for a test case



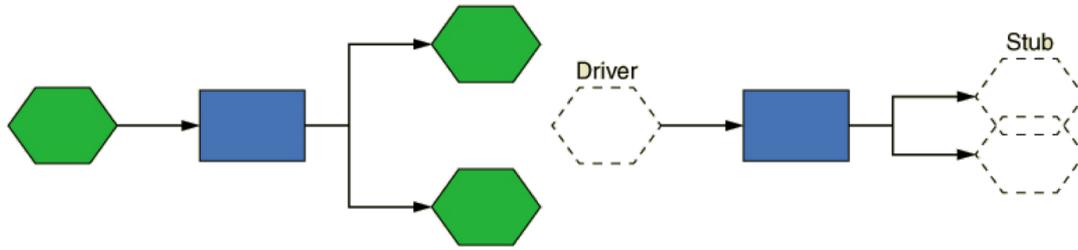
Test Types

Test type	Core process	Need and purpose
Unit testing	Implementation	Software components must perform to the defined requirements and specifications when tested in isolation—for example, a component that incorrectly calculates sales tax amounts in different locations is unacceptable.
Integration testing	Implementation	Software components that perform correctly in isolation must also perform correctly when executed in combination with other components. They must communicate correctly with other components in the system. For example a sales tax component that calculates incorrectly when receiving money amounts in foreign currencies is unacceptable .
System and stress testing	Deployment	A system or subsystem must meet both functional and non-functional requirements. For example an item lookup function in a Sales subsystems retrieves data within 2 seconds when running in isolation, but requires 30 seconds when running within the complete system with a live database.
User acceptance testing	Deployment	Software must not only operate correctly, but must also satisfy the business need and meet all user “ease of use” and “completeness” requirements—for example, a commission system that fails to handle special promotions or a data-entry function with a poorly designed sequence of forms is unacceptable.



Unit Testing

- Unit test – tests of an individual method, class, or component before it is integrated with other software
- Driver – a method or class developed for unit testing that simulates the behavior of a method that sends a message to the method being tested
- Stub – a method or class developed for unit testing that simulates the behavior of a method invoked that hasn't yet been written





Integration Testing

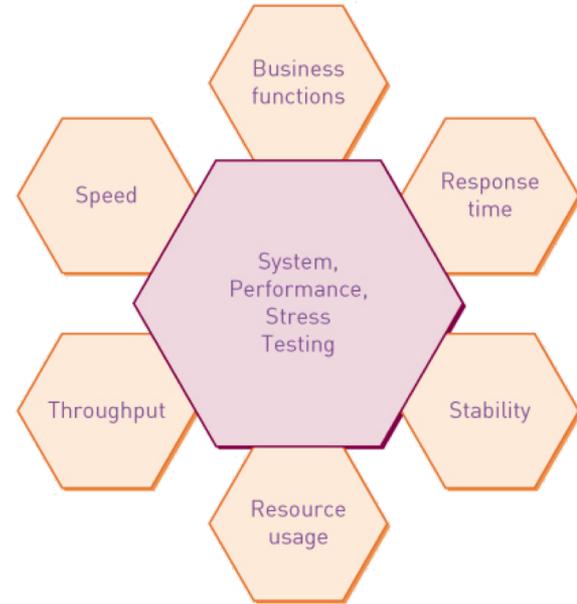
- Integration test – tests of the behaviour of a group of methods, classes, or components
 - Interface incompatibility—For example, one method passes a parameter of the wrong data type to another method
 - Parameter values—A method is passed or returns a value that was unexpected, such as a negative number for a price.
 - Run-time exceptions—A method generates an error, such as “out of memory” or “file already in use,” due to conflicting resource needs
 - Unexpected state interactions—The states of two or more objects interact to cause complex failures, as when an OnlineCart class method operates correctly for all possible Customer object states except one

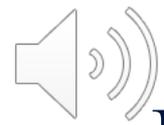




System, Performance, and Stress Testing

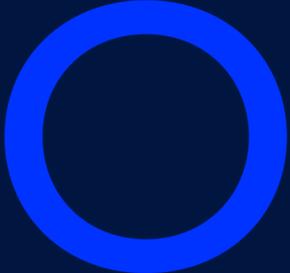
- System test – an integration test of an entire system or independent subsystem
 - Can be performed at the end of each iteration
 - Can be performed more frequently
- Performance test or stress test – an integration and usability test that determines whether a system or subsystem can meet time-based performance criteria
 - Response time – the desired or maximum allowable time limit for software response to a query or update
 - Throughput – the desired or minimum number of queries and transactions that must be processed per minute or hour





User Acceptance Testing (UAT)

- User acceptance test – a system test performed to determine whether the system fulfils user requirements
- May be performed near the end of the project (or at end of later project iterations)
- May be a very formal activity in most development projects.
 - Details of acceptance tests are sometimes included in the request for proposal (RFP) and procurement contract
 - Payments tied to passing tests
- May be informal and integrated into the normal development activities .



Deploying the New System

CSIT883 System Analysis and Project Management



UNIVERSITY
OF WOLLONGONG
AUSTRALIA





Outline

Testing

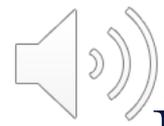
Part I

Deployment Activities

Part II

Managing Implementation and Deployment

Part III



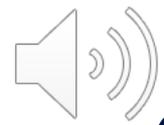
Deployment Activities

- Note system tests, stress tests, and user acceptance tests are considered deployment

Deployment activities

Perform system and stress tests.
Perform user acceptance tests.
Convert existing data.
Build training materials and conduct training.
Configure and set up production environment.
Deploy the solution.

Core processes	Iterations					
	1	2	3	4	5	6
Identify the problem and obtain approval.	[Activity represented by a semi-circle in the first iteration]					
Plan and monitor the project.	[Activity represented by a semi-circle in the first iteration]					
Discover and understand details.	[Activity represented by a semi-circle in the first iteration]					
Design system components.	[Activity represented by a semi-circle in the first iteration]					
Build, test, and integrate system components.	[Activity represented by a semi-circle in the first iteration]					
Complete system tests and deploy the solution.	[Activity represented by a semi-circle in the first iteration]					



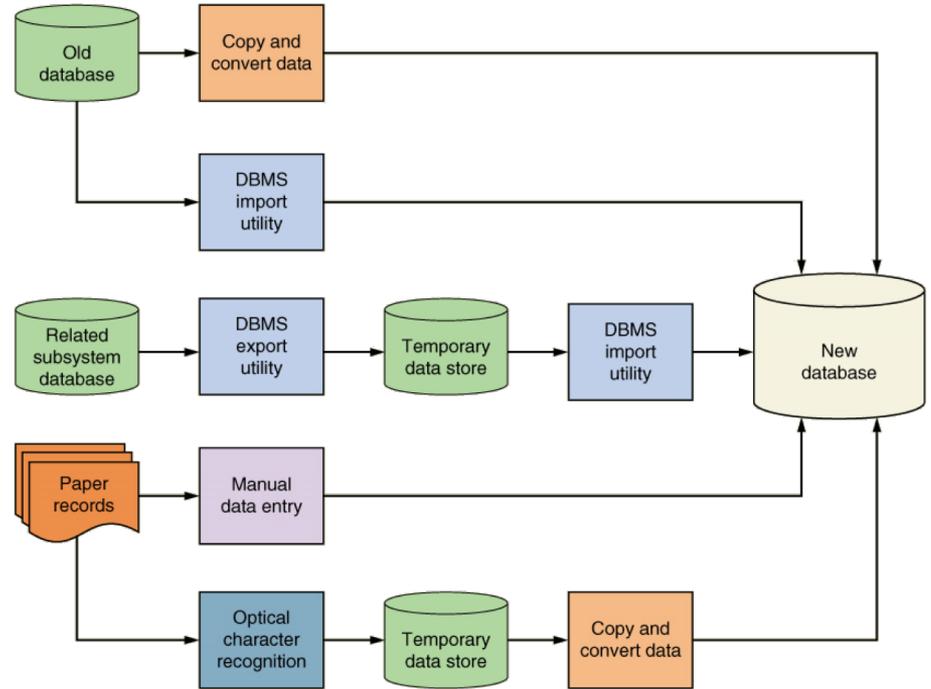
Converting and Initializing Data

- An operational system requires a fully populated database to support ongoing processing
- Data needed at system start-up can be obtained from these sources:
 - Files or databases of a system being replaced
 - Manual records
 - Files or databases from other systems in the organization
 - User feedback during normal system operation



Converting and Initializing Data

- Reuse existing databases
 - Modify or update existing data
- Reload databases
 - Copy and convert the data
 - Export and import data from distinct DBMSs
 - Data entry from paper documents



Example of Complex data conversion



Training Users

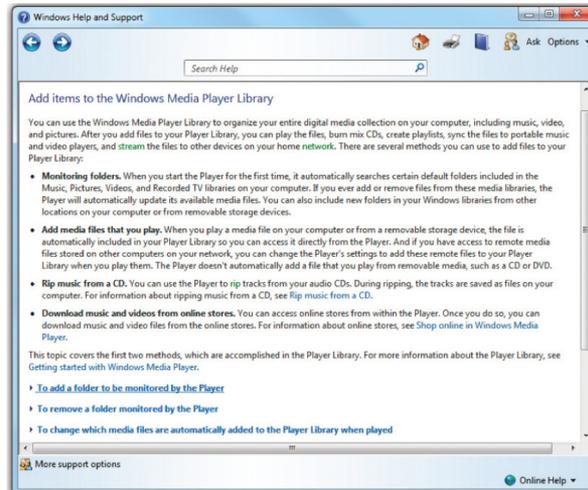
- Training for **end users** must emphasize hands-on use for specific business processes or functions, such as order entry, inventory control, or accounting
 - Widely varying skill and experience levels call for at least some hands-on training, including practice exercises, questions and answers, and one-on-one tutorials
- **System operator** training can be much less formal when the operators aren't end users
 - Experienced computer operators and administrators can learn most or all they need to know by self-study

End-user activities	System operator activities
Creating records or transactions	Starting or stopping the system
Modifying database contents	Querying system status
Generating reports	Backing up data to archive
Querying database	Recovering data from archive
Importing or exporting data	Installing or upgrading software



Training Users

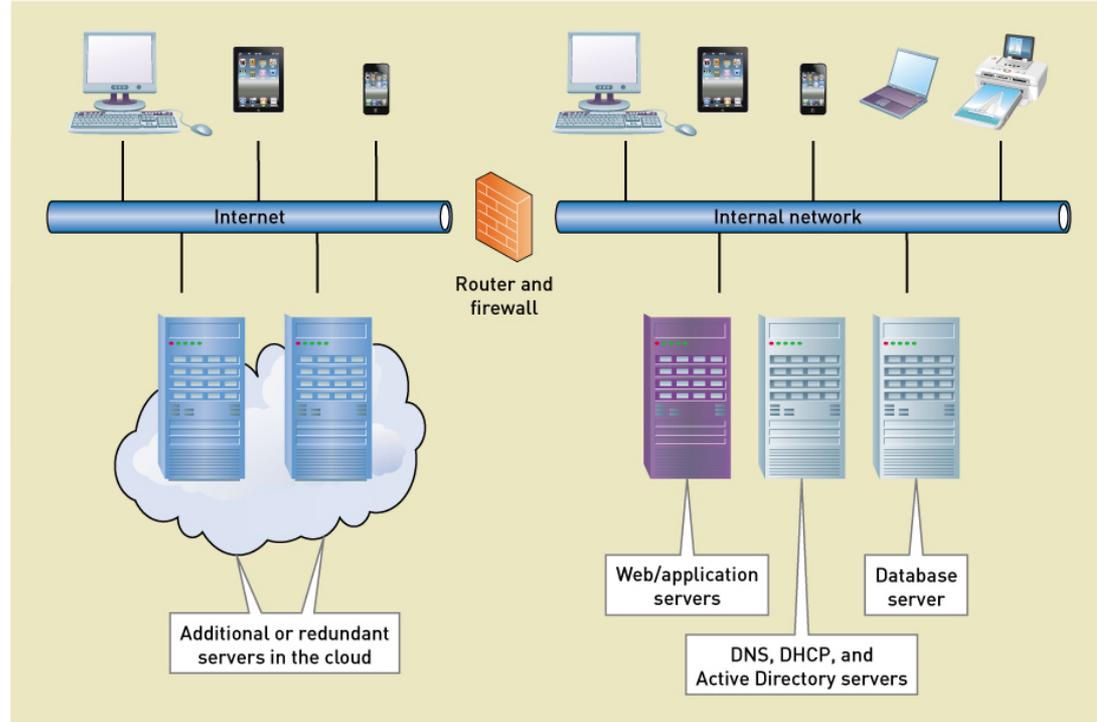
- System Documentation
 - Descriptions of system requirements and architecture to help maintenance and upgrade of the system
- User Documentation
 - How to interact with and use the system for end users and system operators

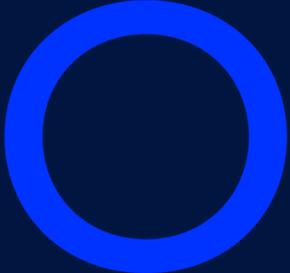




Configuring the Production Environment

- Example:
Infrastructure and clients on a Microsoft .NET application





Deploying the New System

CSIT883 System Analysis and Project Management



UNIVERSITY
OF WOLLONGONG
AUSTRALIA





Outline

Testing

Part I

Deployment Activities

Part II

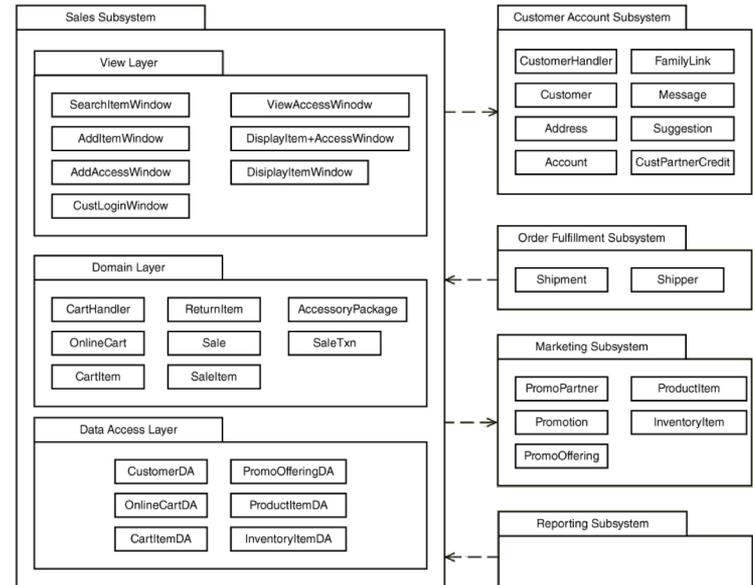
Managing Implementation and Deployment

Part III

Planning and Managing Implementation and Deployment

- Development Order

- Input, process, output (IPO) – a development order that implements input modules first, process modules next, and output modules last

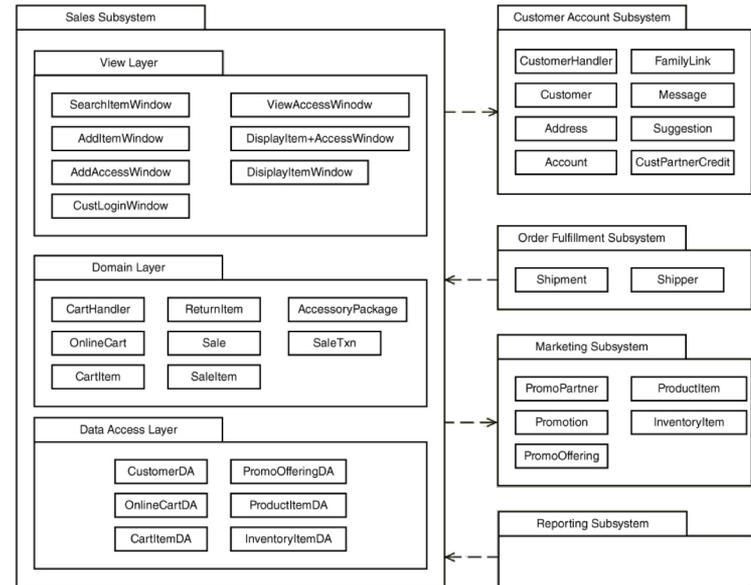


Example: Package diagram showing dependencies, which constrains development order

Planning and Managing Implementation and Deployment

- Development Order

- Input, process, output (IPO) – a development order that implements input modules first, process modules next, and output modules last
- Top-down development – a development order that implements top-level modules first
- Bottom-up development – a development order that implements low-level detailed modules first
- Use-case driven – select specific use cases and order the development based on selected use cases

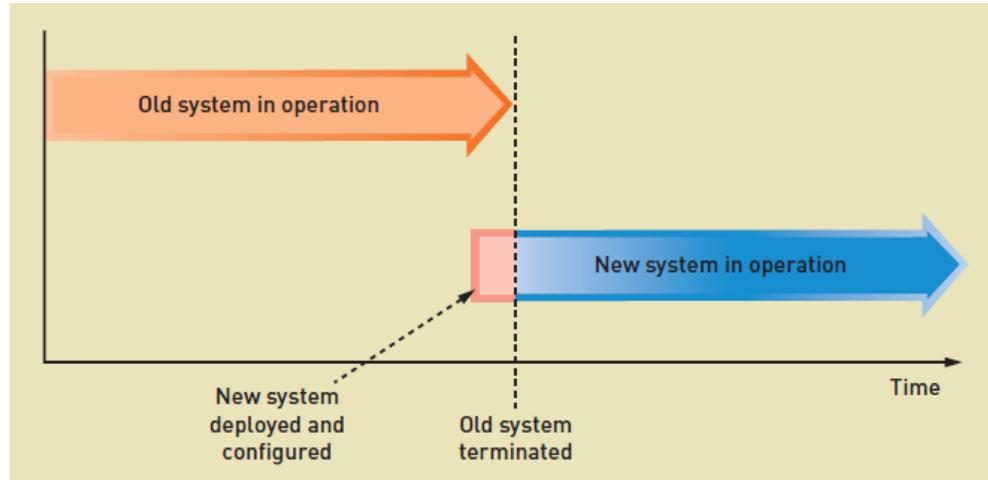


Example: Package diagram showing dependencies, which constrains development order



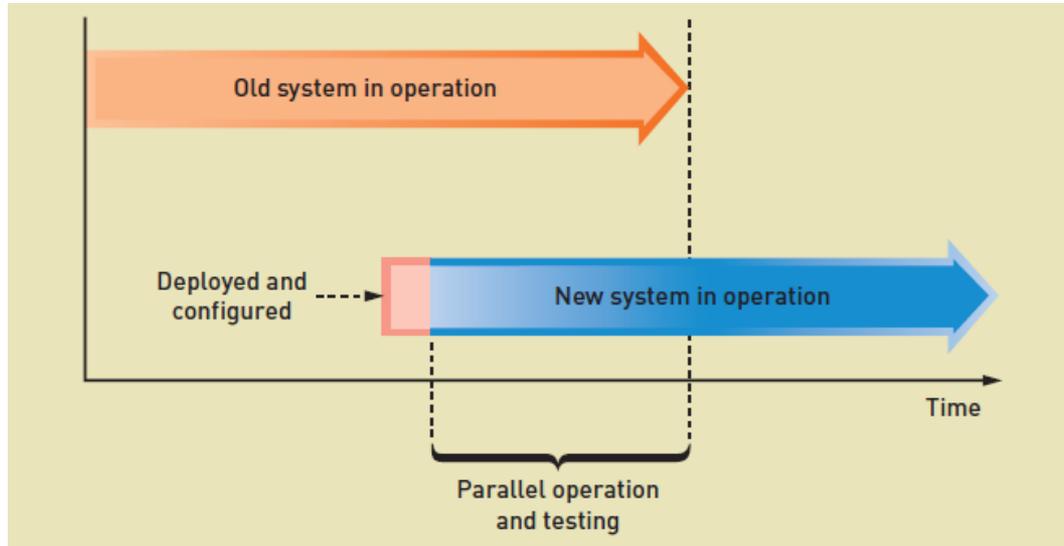
Planning and Managing Implementation and Deployment

- Different approaches: Direct deployment, Parallel deployment, and Phased deployment
- Direct deployment – a deployment method that installs a new system, quickly makes it operational, and immediately turns off any overlapping systems
 - Higher risk, lower cost



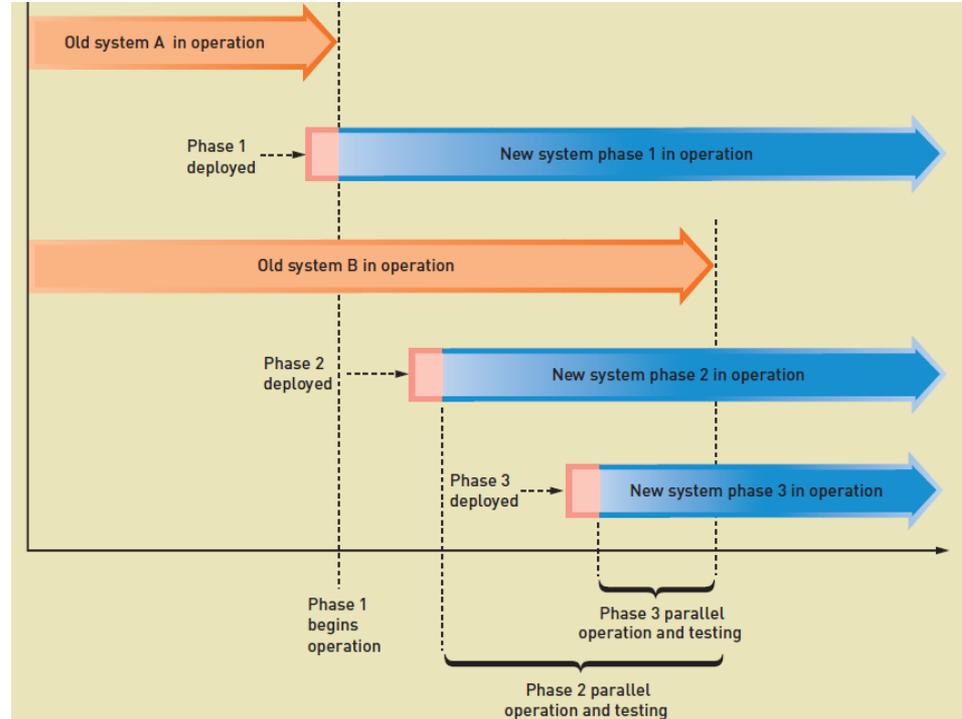
Planning and Managing Implementation and Deployment

- Parallel deployment – a deployment method that operates the old and the new systems for an extended time period
 - Lower risk, higher cost, etc.



Planning and Managing Implementation and Deployment

- Phased deployment
 - a deployment method that installs a new system and makes it operational in a series of steps or phases





Planning and Managing Implementation and Deployment

- Change and Version Control – tools and processes handle the complexity associated with testing and supporting a system through multiple versions
 - Alpha version – a test version that is incomplete but ready for some level of rigorous integration or usability testing
 - Beta version – a test version that is stable enough to be tested by end users over an extended period of time
 - Production version, release version, or production release – a system version that is formally distributed to users or made operational for long-term use
 - Maintenance release – a system update that provides bug fixes and small changes to existing features



Summary

- Implementation and deployment are complex processes because they consist of many interdependent activities.
- Testing is a key activity of implementation and deployment.
- Configuration and change management activities track changes to models and software through multiple system versions.