# Chapter 13
# Making the system operational

Asst.Prof.Dr. Supakit Nootyaskool
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

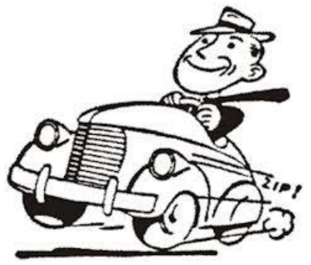
## Outline

- Testing
- Deployment Activities
- Planning and Managing Implementation, Testing, and Deployment
- Putting It All Together—RMO Revisited

## Objective

- Describe implementation and deployment activities
- Describe various types of software tests and explain how and why each is used
- Explain the importance of configuration management, change management, and source code control to the implementation, testing, and deployment of a system
- List various approaches to data conversion and system deployment and describe the advantages and disadvantages of each
- Describe training and user support requirements for new and operational systems

# Overview: Difficult system development



Complex production
Assembly production
Resource efficiently
Minimize construction time
Maximize product quality

Subsequence assembly step controls in automobile.

- Time
- Cost
- Output quality

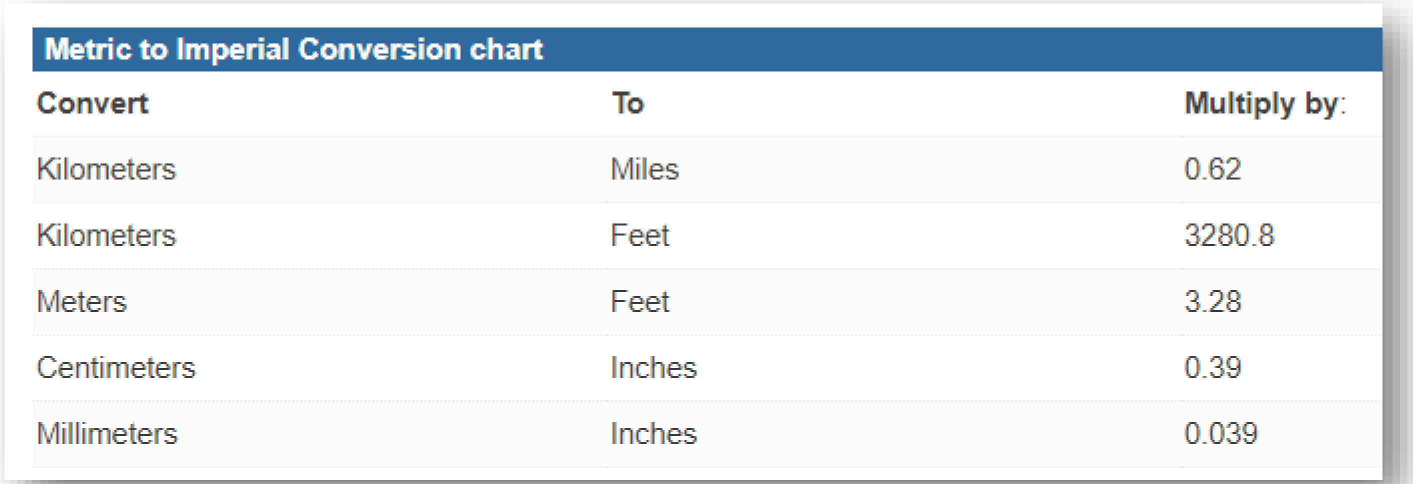
In System software development

- Time
- Cost
- Software Quality

*How is different between automobile production and software production?*

## Writing a program to convert a distance from a mile per second to feet per second

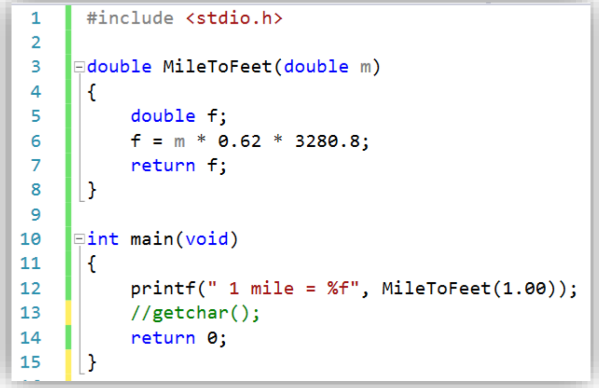
- Write code here !



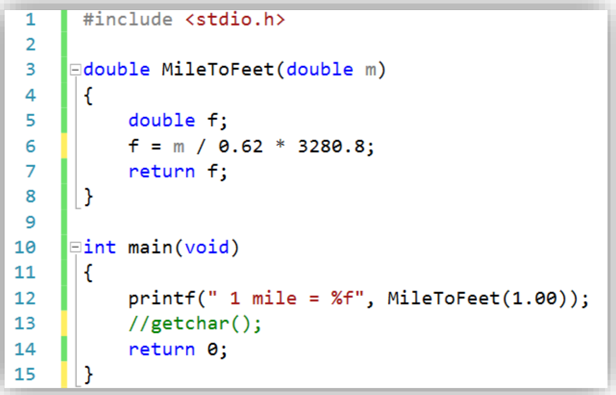
| Metric to Imperial Conversion chart | | |
|---|---|---|
| **Convert** | **To** | **Multiply by:** |
| Kilometers | Miles | 0.62 |
| Kilometers | Feet | 3280.8 |
| Meters | Feet | 3.28 |
| Centimeters | Inches | 0.39 |
| Millimeters | Inches | 0.039 |

# Writing a program to convert a distance from a mile per second to feet per second

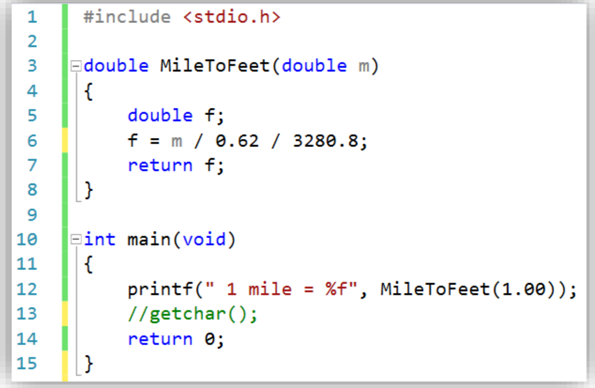
- Which code is correct ?



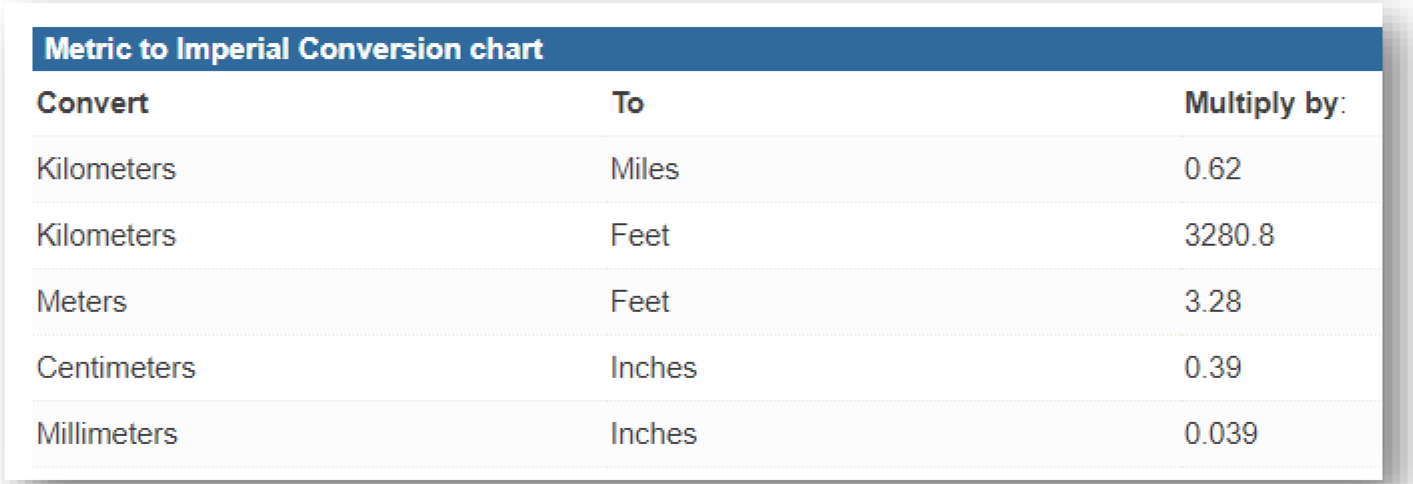
```c
#include <stdio.h>

double MileToFeet(double m)
{
    double f;
    f = m * 0.62 * 3280.8;
    return f;
}

int main(void)
{
    printf(" 1 mile = %f", MileToFeet(1.00));
    //getchar();
    return 0;
}
```



```c
#include <stdio.h>

double MileToFeet(double m)
{
    double f;
    f = m / 0.62 * 3280.8;
    return f;
}

int main(void)
{
    printf(" 1 mile = %f", MileToFeet(1.00));
    //getchar();
    return 0;
}
```



```c
#include <stdio.h>

double MileToFeet(double m)
{
    double f;
    f = m / 0.62 / 3280.8;
    return f;
}

int main(void)
{
    printf(" 1 mile = %f", MileToFeet(1.00));
    //getchar();
    return 0;
}
```

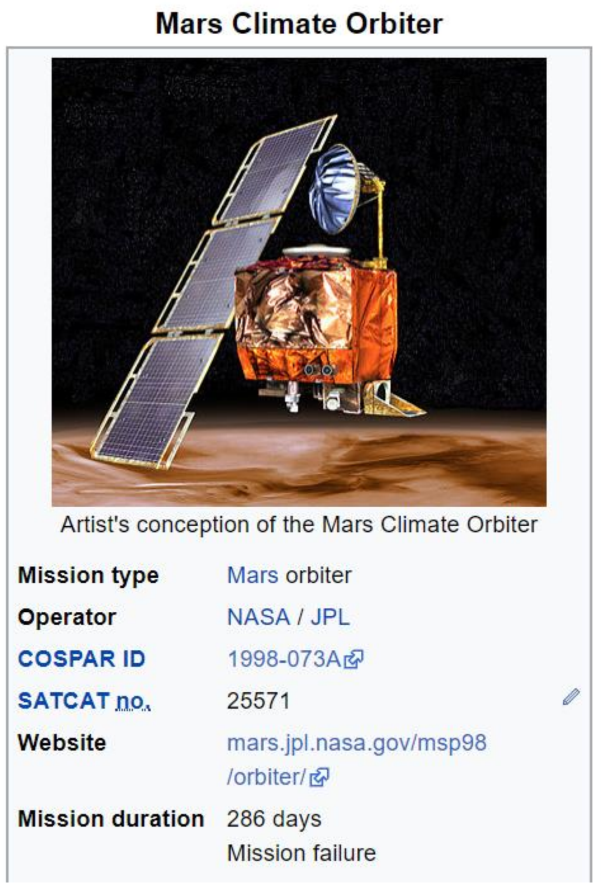


**Metric to Imperial Conversion chart**

| Convert | To | Multiply by: |
|---|---|---|
| Kilometers | Miles | 0.62 |
| Kilometers | Feet | 3280.8 |
| Meters | Feet | 3.28 |
| Centimeters | Inches | 0.39 |
| Millimeters | Inches | 0.039 |

## Why Product Testing is Importance?

- In September 1999, the Mars Climate Orbiter crashed into the planet instead of reaching a safe orbit. A report by a NASA investigation board stated that the main reason for the loss of the spacecraft was a failure to convert measurements of rocket thrusts from English units to metric units in a section of ground-based navigation related mission software.

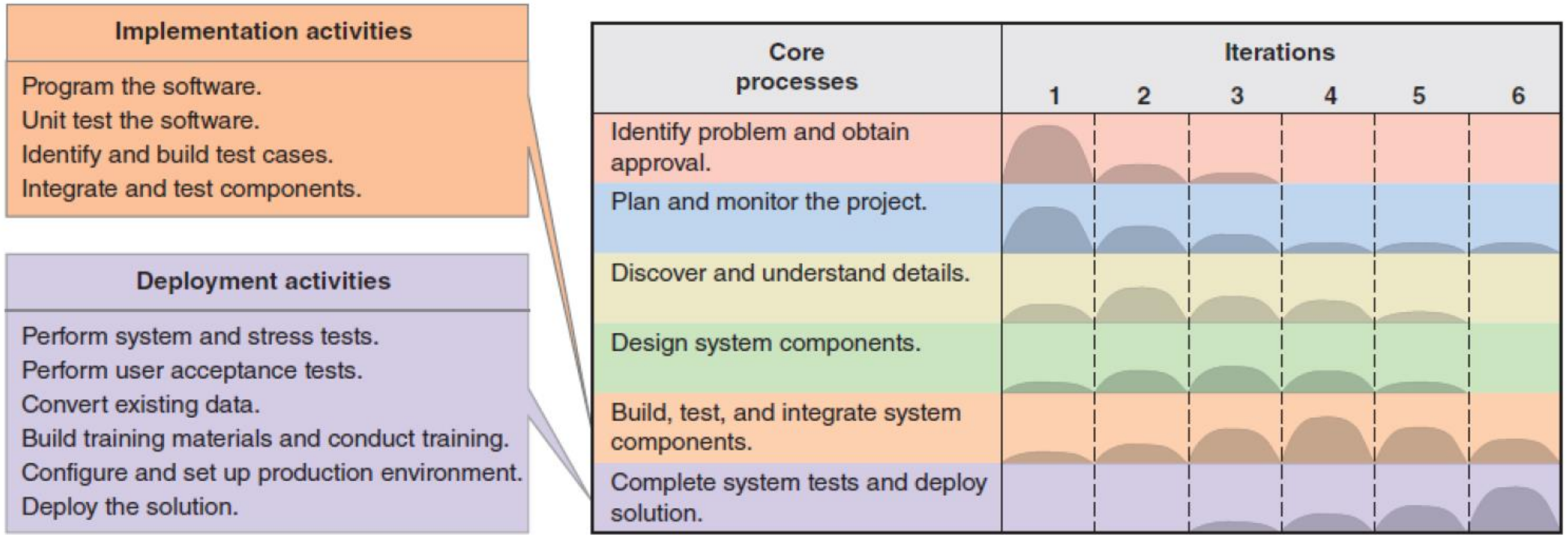


**Mars Climate Orbiter**

Artist's conception of the Mars Climate Orbiter

| | |
|---|---|
| **Mission type** | Mars orbiter |
| **Operator** | NASA / JPL |
| **COSPAR ID** | 1998-073A |
| **SATCAT no.** | 25571 |
| **Website** | mars.jpl.nasa.gov/msp98/orbiter/ |
| **Mission duration** | 286 days<br>Mission failure |

## Implementation and Deployment activities



**Implementation activities**

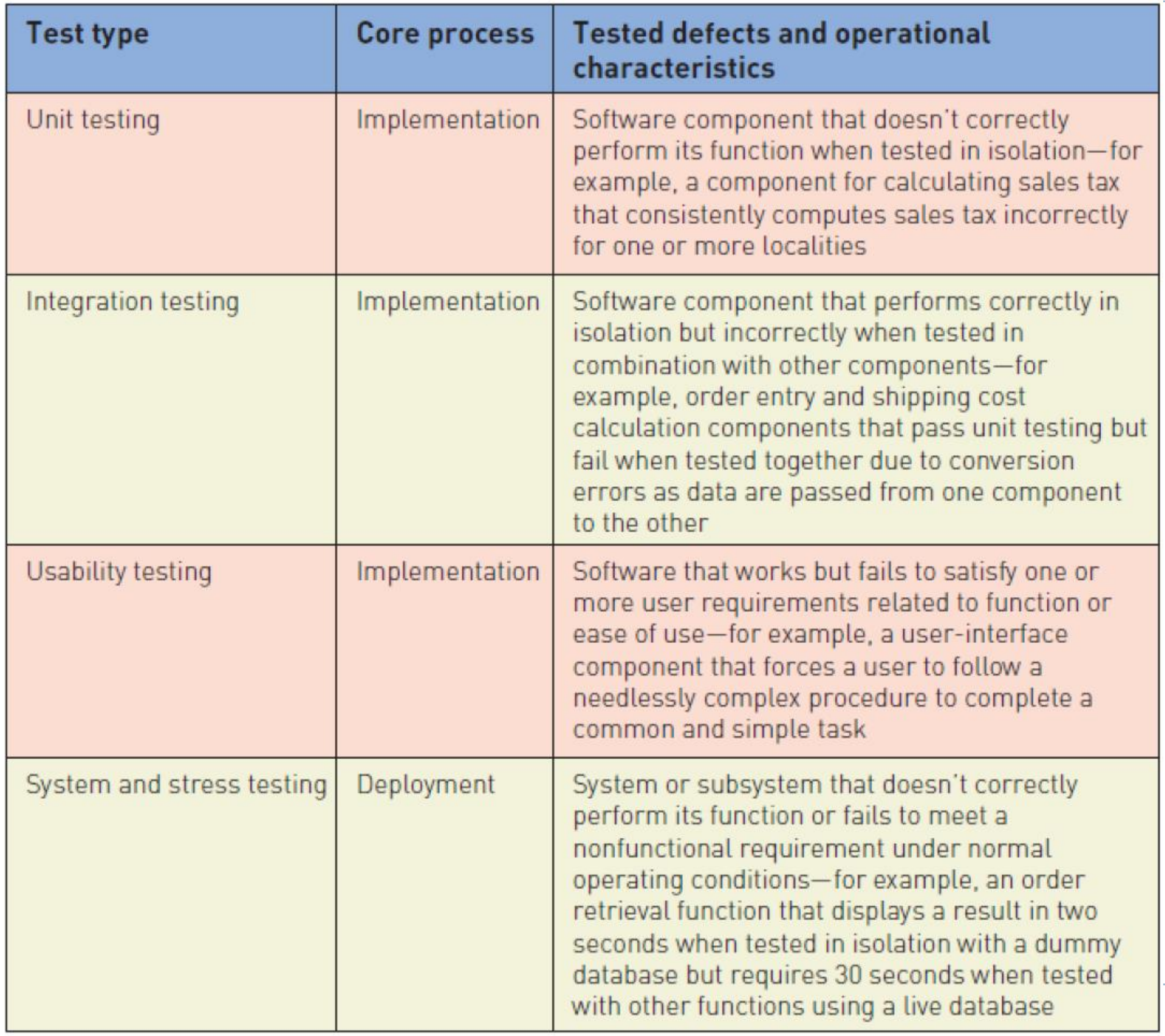
Program the software.
Unit test the software.
Identify and build test cases.
Integrate and test components.

**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 problem and obtain approval.
Plan and monitor the project.
Discover and understand details.
Design system components.
Build, test, and integrate system components.
Complete system tests and deploy solution.

# Test types



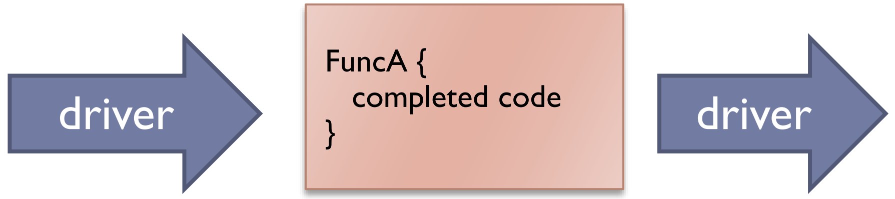
| Test type | Core process | Tested defects and operational characteristics |
|---|---|---|
| Unit testing | Implementation | Software component that doesn't correctly perform its function when tested in isolation—for example, a component for calculating sales tax that consistently computes sales tax incorrectly for one or more localities |
| Integration testing | Implementation | Software component that performs correctly in isolation but incorrectly when tested in combination with other components—for example, order entry and shipping cost calculation components that pass unit testing but fail when tested together due to conversion errors as data are passed from one component to the other |
| Usability testing | Implementation | Software that works but fails to satisfy one or more user requirements related to function or ease of use—for example, a user-interface component that forces a user to follow a needlessly complex procedure to complete a common and simple task |
| System and stress testing | Deployment | System or subsystem that doesn't correctly perform its function or fails to meet a nonfunctional requirement under normal operating conditions—for example, an order retrieval function that displays a result in two seconds when tested in isolation with a dummy database but requires 30 seconds when tested with other functions using a live database |

## Testing concepts

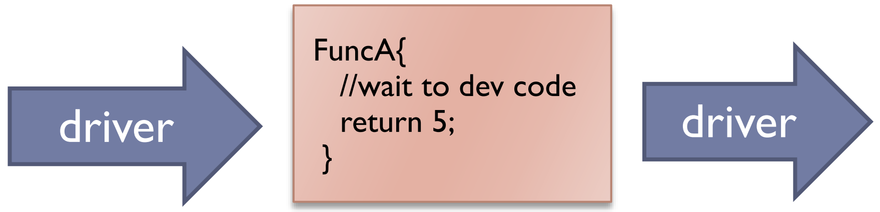
- Testing – the process of examining a component, subsystem, or system to determine its operational characteristics and whether it contains any defects
- Test case – a formal description of a starting state, one or more events to which the software must respond, and the expected response or ending state
  - 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

## Unit testing

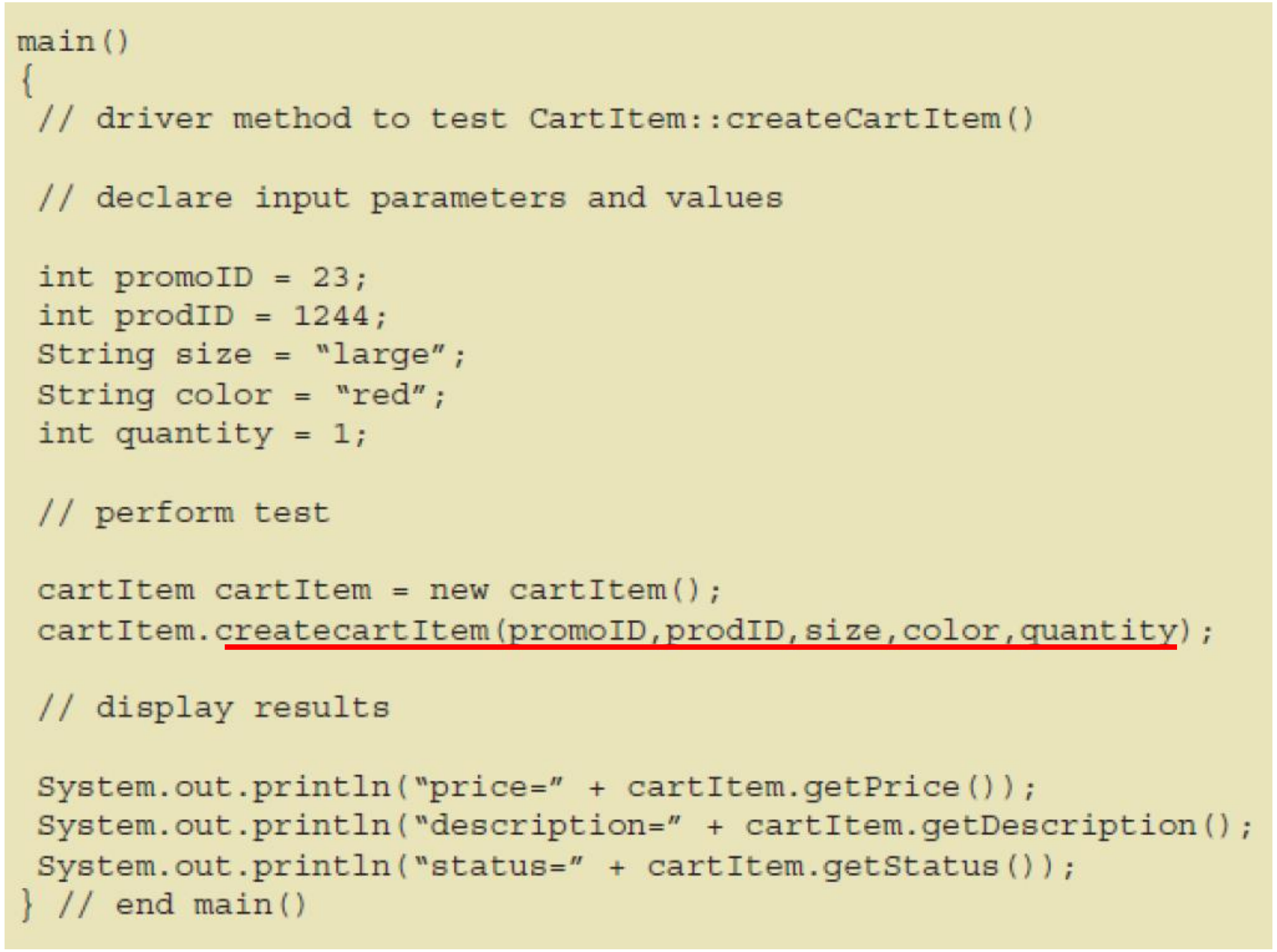
- Unit test – tests of an individual method, class, or component before it is integrated with other software
- Driver – a method or class developed for unit testing that simulates the behavior of a method that sends a message to the method being tested



- Stub – a method or class developed for unit testing that simulates the behavior of a method invoked that hasn't yet been written



## Unit testing
## Driver to test createCartItem method



```
main()
{
 // driver method to test CartItem::createCartItem()

 // declare input parameters and values

 int promoID = 23;
 int prodID = 1244;
 String size = "large";
 String color = "red";
 int quantity = 1;

 // perform test

 cartItem cartItem = new cartItem();
 cartItem.createcartItem(promoID,prodID,size,color,quantity);

 // display results

 System.out.println("price=" + cartItem.getPrice());
 System.out.println("description=" + cartItem.getDescription();
 System.out.println("status=" + cartItem.getStatus());
} // end main()
```

## Unit testing
## Some stub modules used by createCartItem

```
float getPrice()
{
 // stub method for CatalogProduct::getPrice()

 return(24.95);
} // end getPrice()

String getDescription()
{
 // stub method for Product::getDescription()

 return("mens khaki slacks");
} // end getDescription()

String updateQty(int decrement)
{
 // stub method for InventoryItem::updateQty()

 return("OK");
} // end updateQty()
```

## Integration testing

- Integration test – tests of the behavior 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

## Integration testing

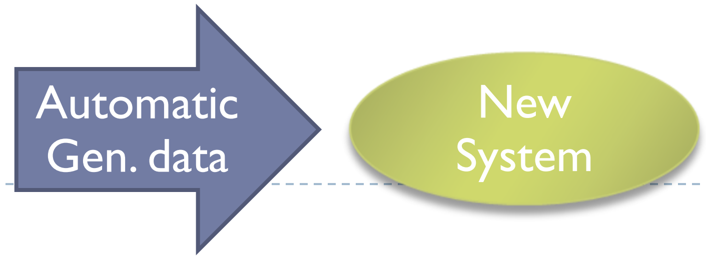
- Integration testing of object-oriented software is very complex because an object-oriented program consists of a set of interacting objects
  - Methods can be (and usually are) called by many other methods, and the calling methods may be distributed across many classes.
  - Classes may inherit methods and state variables from other classes.
  - The specific method to be called is dynamically determined at run time based on the number and type of message parameters.
  - Objects can retain internal variable values (i.e., the object state) between calls. The response to two identical calls may be different due to state changes that result from the first call or occur between calls.

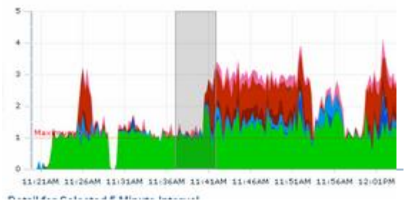
## Usability testing

- Usability test – a test to determine whether a method, class, subsystem, or system meets user requirements
- Many usability tests are required because they involve functional and non-functional requirements
- Most common type evaluates functional requirements, use case by use case
  - Can be completed in each iteration as use cases are implemented
  - Can test ease of learning and ease of use
  - Can test whether results match actual requirements
  - Key type of feedback from users throughout project

## System, performance, and stress testing

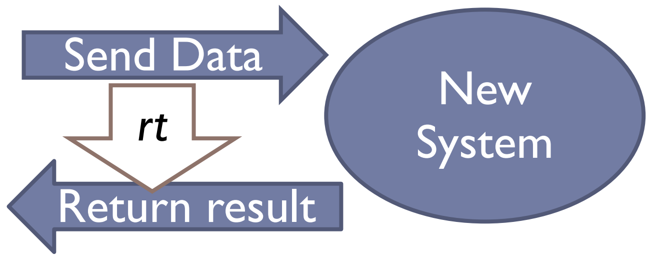
- System test – an integration test of an entire system or independent subsystem
- Can be performed at the end of each iteration
- Can be performed more frequently
- Build and smoke test – a system test that is performed daily or several times a week
  - The system is completely compiled and linked (built), and a battery of tests is executed to see whether anything malfunctions in an obvious way ("smokes")
  - Automated testing tools are used. Catches any problems that may have come up since the last system test





## System, performance, and stress testing

- 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



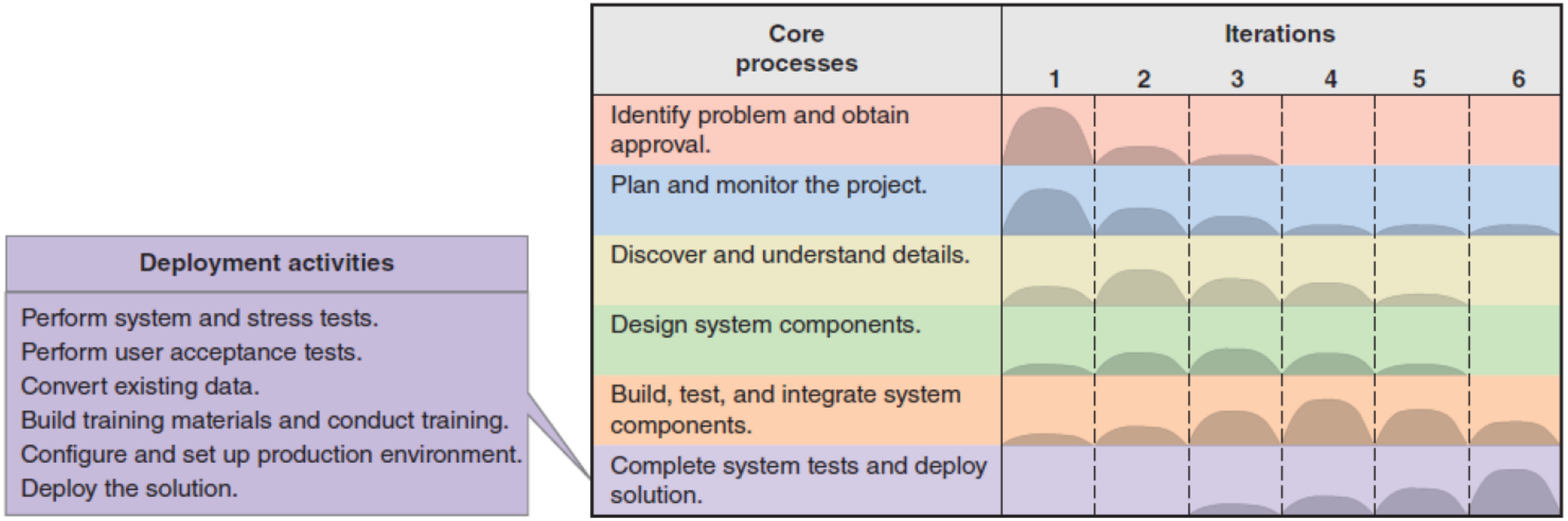
  - Throughput – the desired or minimum number of queries and transactions that must be processed per minute or hour

## User acceptance testing

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

## Deployment activities

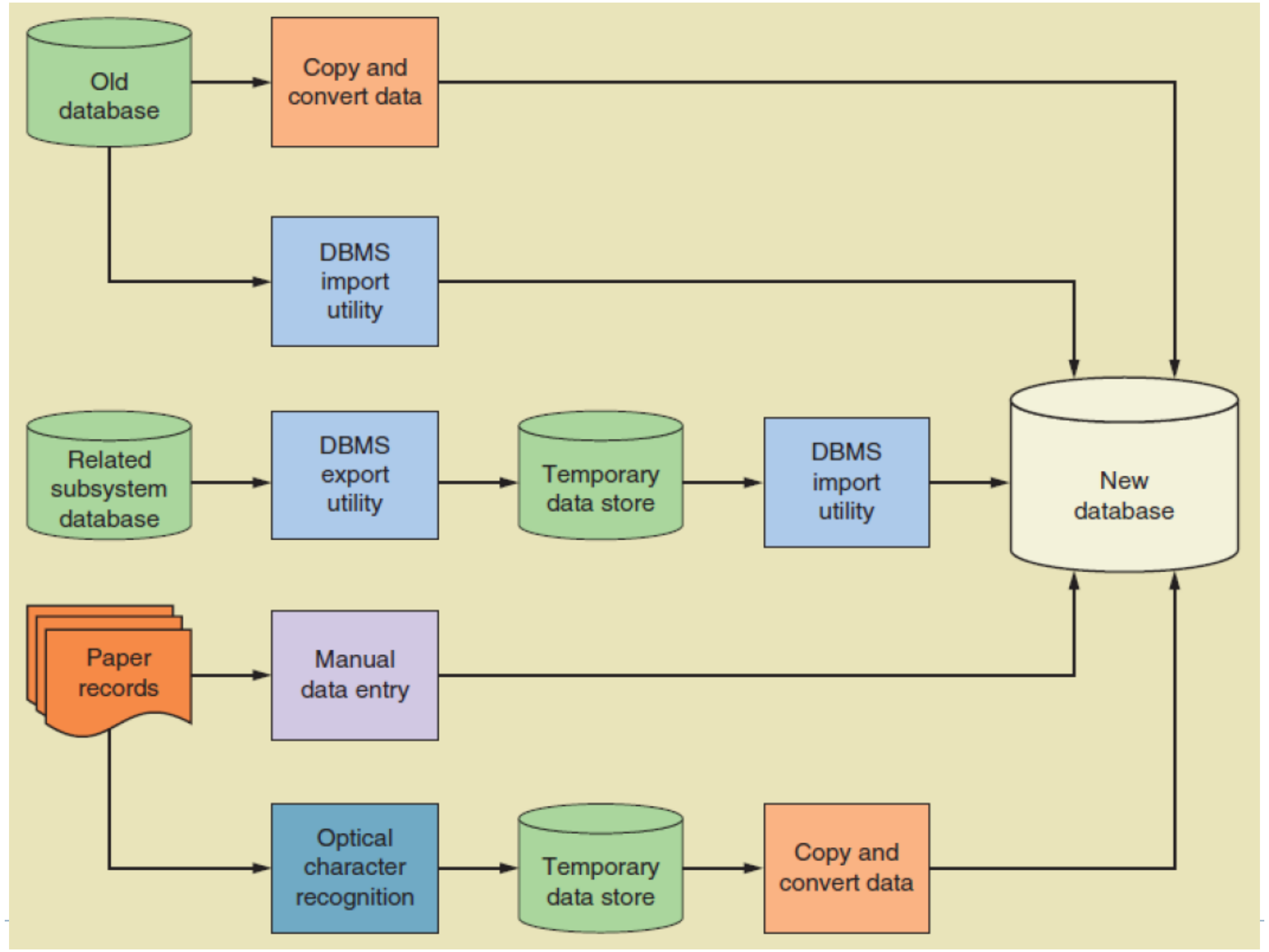
- Note system tests, stress tests, and user acceptance tests are considered deployment


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 problem and obtain approval.
Plan and monitor the project.
Discover and understand details.
Design system components.
Build, test, and integrate system components.
Complete system tests and deploy solution.

## Converting and initializing data

- An operational system requires a fully populated database to support ongoing processing
- Data needed at system startup 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

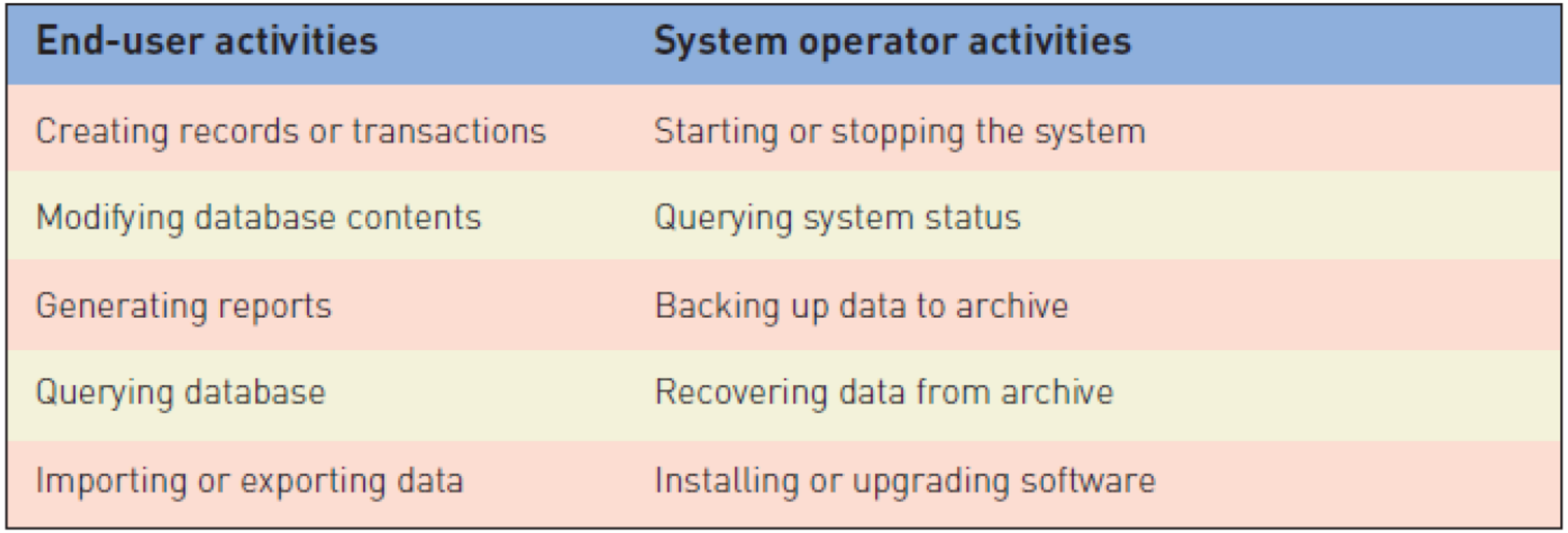
# Converting and Initialing data complex data conversion example



# Training users

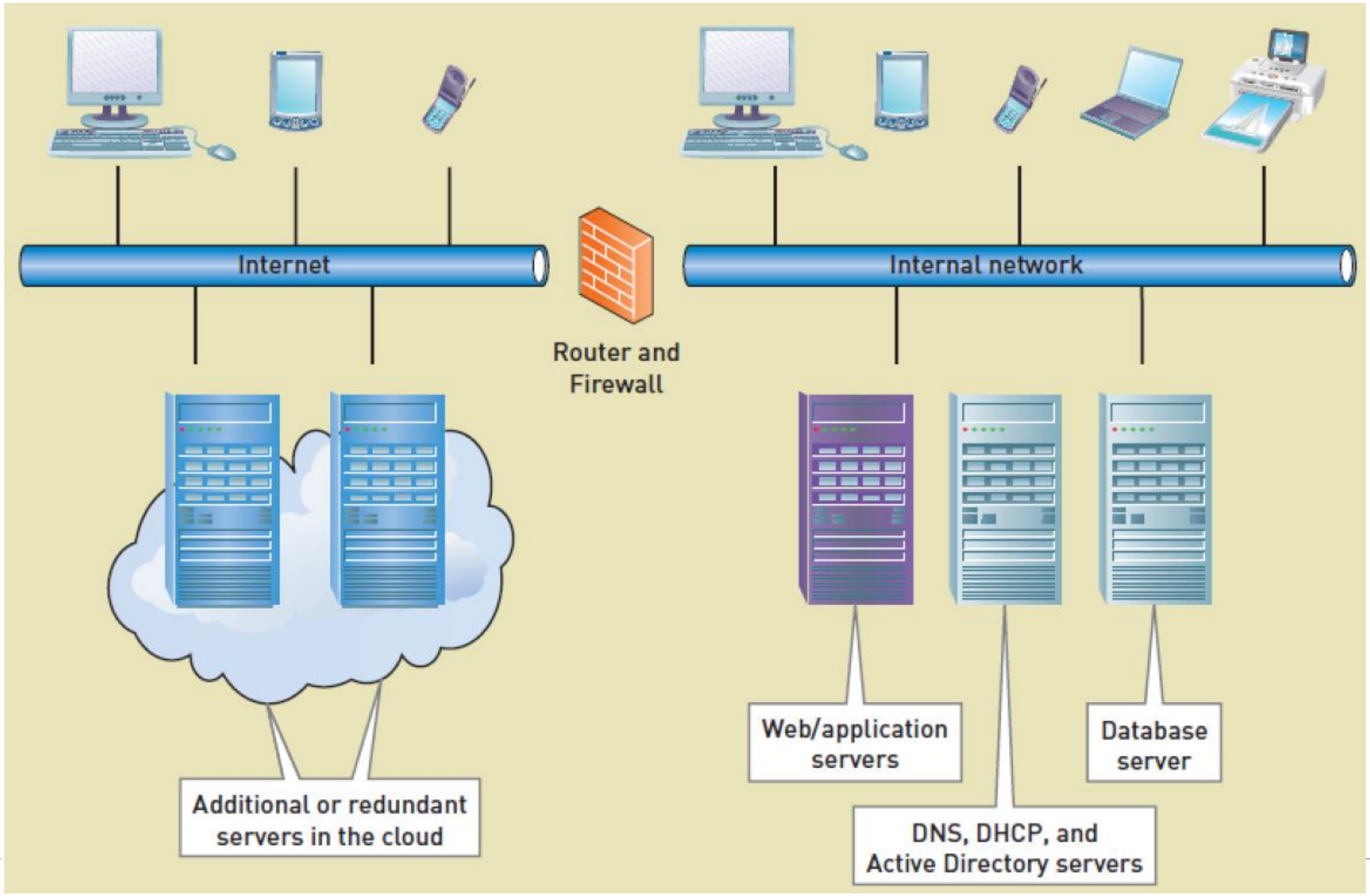
- Training is needed for end users and system operators
- Training for end users must emphasize hands-on use for specific business processes or functions, such as order entry, inventory control, or accounting
  - Widely varying skill and experience levels call for at least some hands-on training, including 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

## Training users



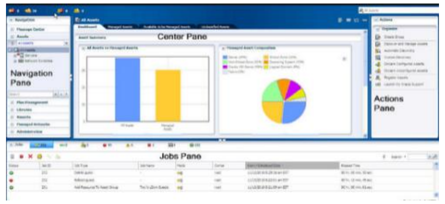
| End-user activities | System operator activities |
|---|---|
| Creating records or transactions | Starting or stopping the system |
| Modifying database contents | Querying system status |
| Generating reports | Backing up data to archive |
| Querying database | Recovering data from archive |
| Importing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

## Configuring the production environment


Internet
Internal network
Router and Firewall
Additional or redundant servers in the cloud
Web/application servers
DNS, DHCP, and Active Directory servers
Database server

## Planning and managing
## Implementation, testing 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
  - Use stubs for testing
- Bottom-up development – a development order that implements low-level detailed modules first
  - Use drivers for testing

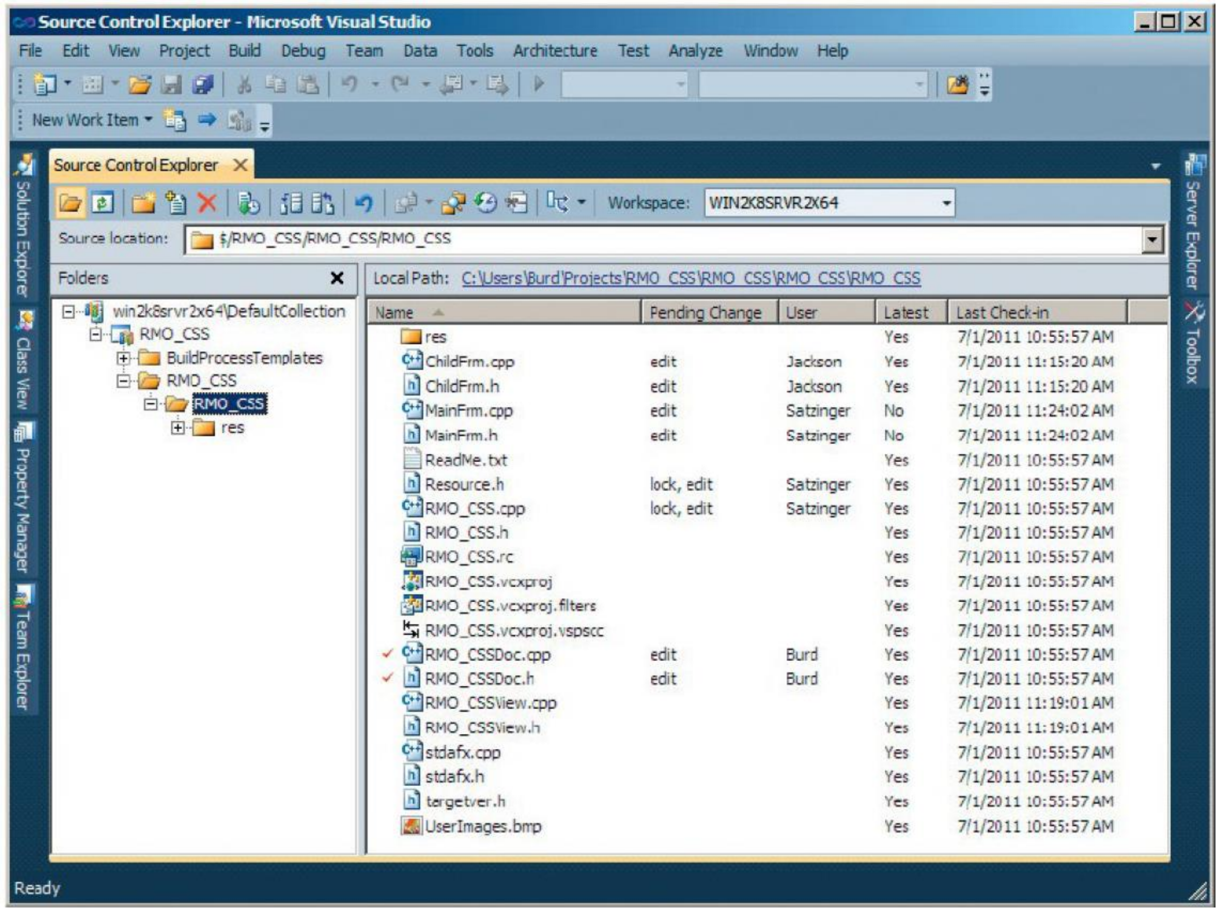


# Planning and managing Implementation, testing and deployment

- Source code control
- An automated tool for tracking source code files and controlling changes to those files
  - A programmer checks out a file in read-only mode when he or she wants to examine the code without making changes (e.g., to examine a module's interfaces to other modules)
  - When a programmer needs to make changes to a file, he or she checks out the file in read/write mode
  - The SCCS allows only one programmer at a time to check out a file in read/write mode.

# Source code control system (SCCS)

- Git.
- Perforce Helix Core.
- Subversion.
- ClearCase.
- Team Foundation Server.


Source Control Explorer - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
New Work Item
Source Control Explorer
Solution Explorer
Class View
Property Manager
Team Explorer
Server Explorer
Toolbox
Folders
Ready

Workspace: WIN2K8SRVR2X64

Source location: $/RMO_CSS/RMO_CSS/RMO_CSS

Local Path: C:\Users\Burd'Projects'RMO_CSS\RMO_CSS\RMO_CSS\RMO_CSS

```
win2k8srvr2x64\DefaultCollection
  RMO_CSS
    BuildProcessTemplates
    RMO_CSS
      RMO_CSS
        res
```

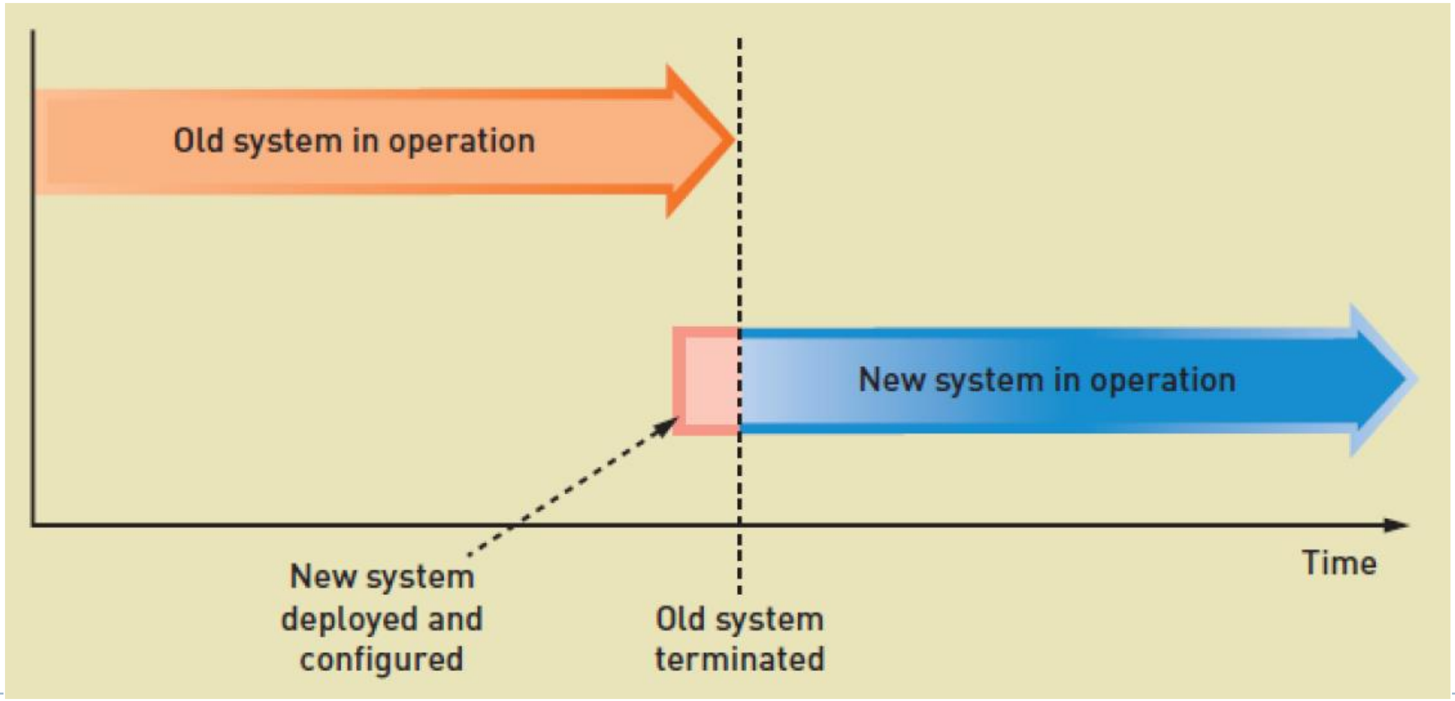
| Name | Pending Change | User | Latest | Last Check-in |
|---|---|---|---|---|
| res | | | Yes | 7/1/2011 10:55:57 AM |
| ChildFrm.cpp | edit | Jackson | Yes | 7/1/2011 11:15:20 AM |
| ChildFrm.h | edit | Jackson | Yes | 7/1/2011 11:15:20 AM |
| MainFrm.cpp | edit | Satzinger | No | 7/1/2011 11:24:02 AM |
| MainFrm.h | edit | Satzinger | No | 7/1/2011 11:24:02 AM |
| ReadMe.txt | | | Yes | 7/1/2011 10:55:57 AM |
| Resource.h | lock, edit | Satzinger | Yes | 7/1/2011 10:55:57 AM |
| RMO_CSS.cpp | lock, edit | Satzinger | Yes | 7/1/2011 10:55:57 AM |
| RMO_CSS.h | | | Yes | 7/1/2011 10:55:57 AM |
| RMO_CSS.rc | | | Yes | 7/1/2011 10:55:57 AM |
| RMO_CSS.vcxproj | | | Yes | 7/1/2011 10:55:57 AM |
| RMO_CSS.vcxproj.filters | | | Yes | 7/1/2011 10:55:57 AM |
| RMO_CSS.vcxproj.vspscc | | | Yes | 7/1/2011 10:55:57 AM |
| ✓ RMO_CSSDoc.cpp | edit | Burd | Yes | 7/1/2011 10:55:57 AM |
| ✓ RMO_CSSDoc.h | edit | Burd | Yes | 7/1/2011 10:55:57 AM |
| RMO_CSSView.cpp | | | Yes | 7/1/2011 11:19:01 AM |
| RMO_CSSView.h | | | Yes | 7/1/2011 11:19:01 AM |
| stdafx.cpp | | | Yes | 7/1/2011 10:55:57 AM |
| stdafx.h | | | Yes | 7/1/2011 10:55:57 AM |
| targetver.h | | | Yes | 7/1/2011 10:55:57 AM |
| UserImages.bmp | | | Yes | 7/1/2011 10:55:57 AM |

## Planning and managing Implementation, Testing and Deployment

- Packaging, installing, and deploying components
  - Issues to consider when planning
    - Incurring costs of operating both systems in parallel
    - Detecting and correcting errors in the new system
    - Potentially disrupting the company and its IS operations
    - Training personnel and familiarizing customers with new procedures
  - Different approaches
    - Direct deployment
    - Parallel deployment
    - Phased deployment

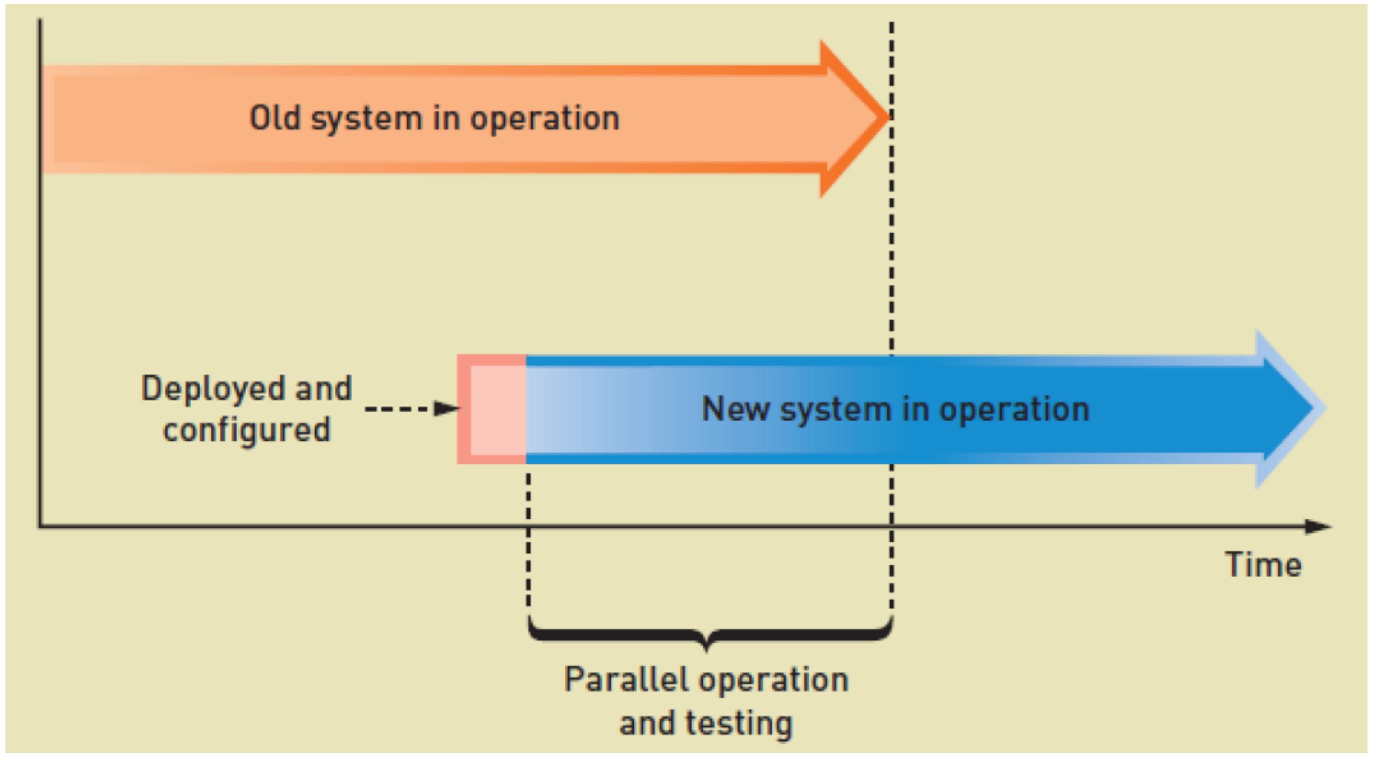
## Planning and managing Implementation, Testing and Deployment

- 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



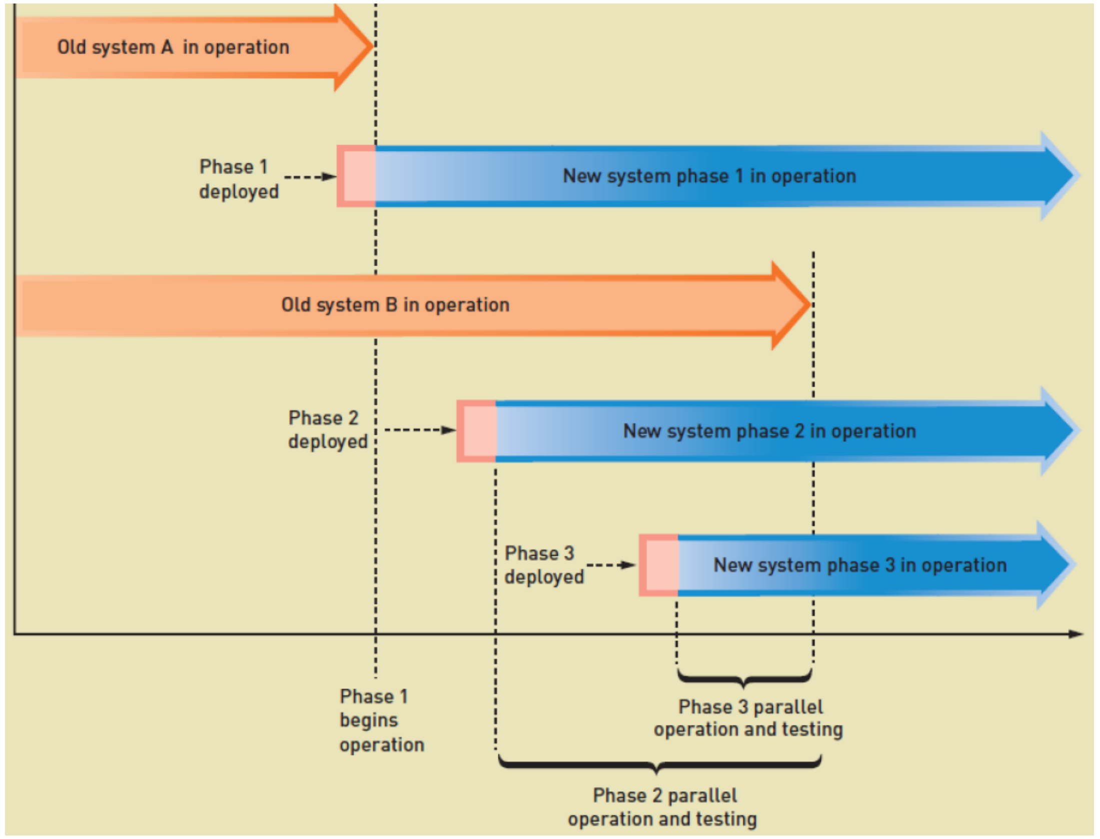
## Planning and managing Implementation, Testing and Deployment

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



## Planning and managing Implementation, Testing and Deployment

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


Old system A in operation
Phase 1 deployed
New system phase 1 in operation
Old system B in operation
Phase 2 deployed
New system phase 2 in operation
Phase 3 deployed
New system phase 3 in operation
Phase 1 begins operation
Phase 3 parallel operation and testing
Phase 2 parallel operation and testing

# Planning and managing Implementation, Testing and Deployment

- Submitting Error Reports and Change Requests
  - Standard reporting methods
  - Review of requests by a project manager or change control committee
  - For operational systems, extensive planning for design and implementation
- Implementing a Change
  - Identify what parts of the system must be changed
  - Secure resources (such as personnel) to implement the change
  - Schedule design and implementation activities
  - Develop test criteria and a testing plan for the changed system

## Planning and managing Implementation, Testing 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

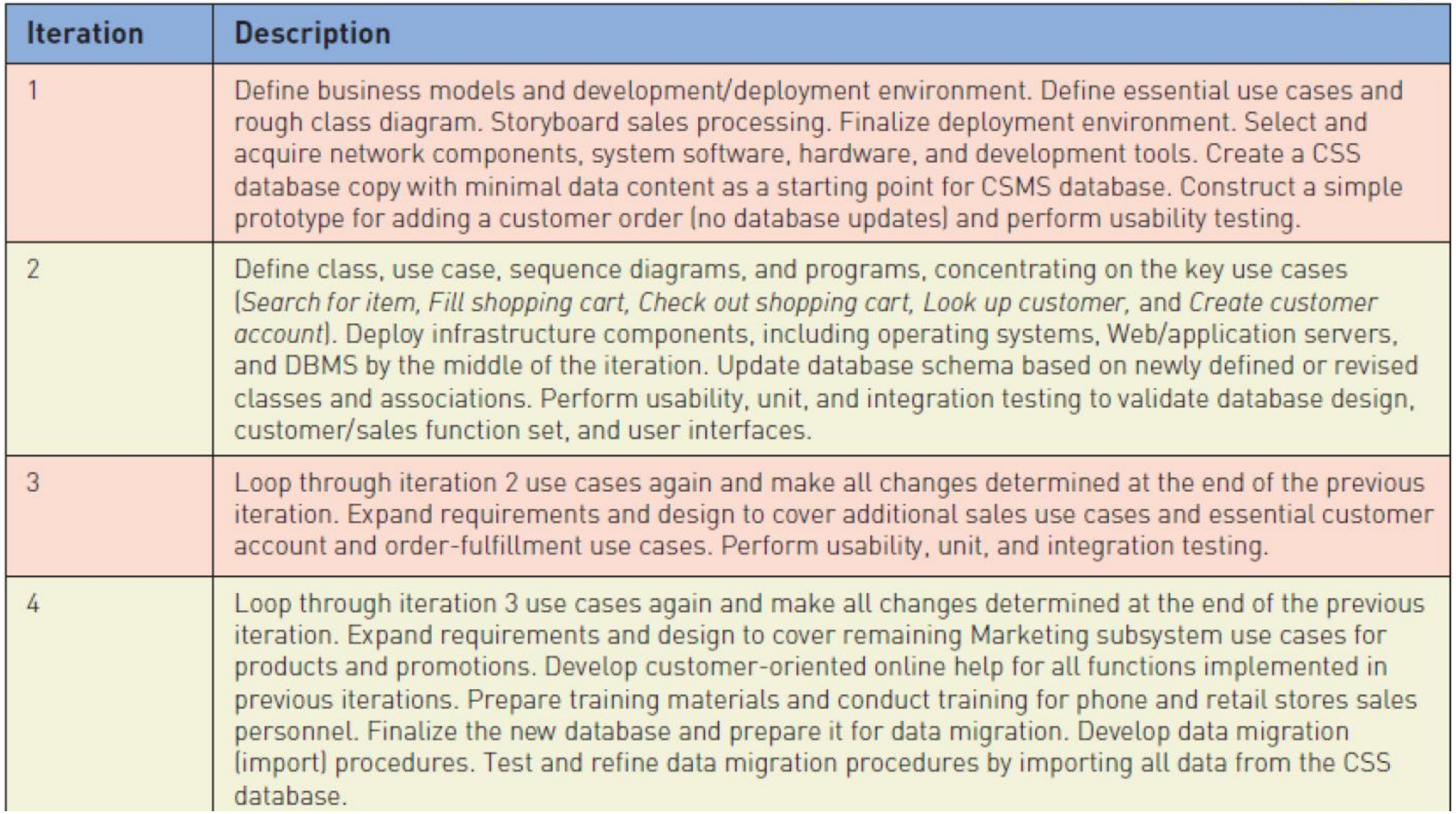
## RMO CSMS system revisited

- Upgrade or Replace?
  - The current infrastructure is near capacity.
  - RMO expects to save money by having an external vendor host the CSMS
  - Existing CSS programs and Web interfaces are a hodgepodge developed over 15 years
  - Current system software is several versions out of date
  - Infrastructure that supports the current CSS can be repurposed to expand SCM capacity
- Therefore RMO decided to Replace

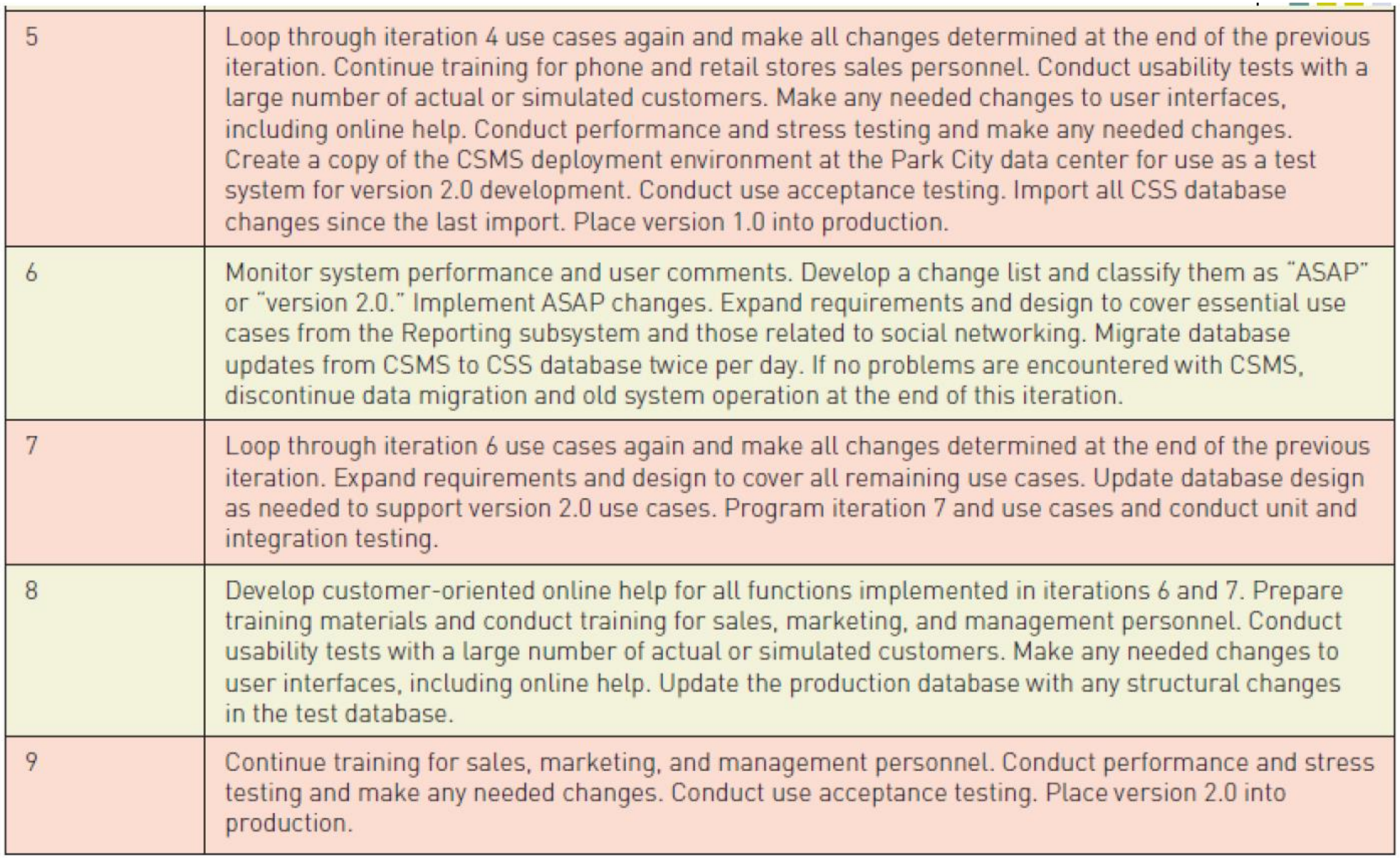
## RMO CSMS system revisited

- Phased Deployment to Minimize Risk
  - Deploy in two versions
- Database Development and Data Conversion
  - New database built and data migrated before deploying version 1, in iterations
- Development Order
  - Start with the higher risk Sales subsystem and customer facing Order fulfillment subsystem
- Documentation and Training
  - Spread throughout later iterations for both versions

## RMO CSMS Iteration plan (part1)



| Iteration | Description |
|---|---|
| 1 | Define business models and development/deployment environment. Define essential use cases and rough class diagram. Storyboard sales processing. Finalize deployment environment. Select and acquire network components, system software, hardware, and development tools. Create a CSS database copy with minimal data content as a starting point for CSMS database. Construct a simple prototype for adding a customer order (no database updates) and perform usability testing. |
| 2 | Define class, use case, sequence diagrams, and programs, concentrating on the key use cases (*Search for item, Fill shopping cart, Check out shopping cart, Look up customer,* and *Create customer account*). Deploy infrastructure components, including operating systems, Web/application servers, and DBMS by the middle of the iteration. Update database schema based on newly defined or revised classes and associations. Perform usability, unit, and integration testing to validate database design, customer/sales function set, and user interfaces. |
| 3 | Loop through iteration 2 use cases again and make all changes determined at the end of the previous iteration. Expand requirements and design to cover additional sales use cases and essential customer account and order-fulfillment use cases. Perform usability, unit, and integration testing. |
| 4 | Loop through iteration 3 use cases again and make all changes determined at the end of the previous iteration. Expand requirements and design to cover remaining Marketing subsystem use cases for products and promotions. Develop customer-oriented online help for all functions implemented in previous iterations. Prepare training materials and conduct training for phone and retail stores sales personnel. Finalize the new database and prepare it for data migration. Develop data migration (import) procedures. Test and refine data migration procedures by importing all data from the CSS database. |



| | |
|---|---|
| 5 | Loop through iteration 4 use cases again and make all changes determined at the end of the previous iteration. Continue training for phone and retail stores sales personnel. Conduct usability tests with a large number of actual or simulated customers. Make any needed changes to user interfaces, including online help. Conduct performance and stress testing and make any needed changes. Create a copy of the CSMS deployment environment at the Park City data center for use as a test system for version 2.0 development. Conduct use acceptance testing. Import all CSS database changes since the last import. Place version 1.0 into production. |
| 6 | Monitor system performance and user comments. Develop a change list and classify them as "ASAP" or "version 2.0." Implement ASAP changes. Expand requirements and design to cover essential use cases from the Reporting subsystem and those related to social networking. Migrate database updates from CSMS to CSS database twice per day. If no problems are encountered with CSMS, discontinue data migration and old system operation at the end of this iteration. |
| 7 | Loop through iteration 6 use cases again and make all changes determined at the end of the previous iteration. Expand requirements and design to cover all remaining use cases. Update database design as needed to support version 2.0 use cases. Program iteration 7 and use cases and conduct unit and integration testing. |
| 8 | Develop customer-oriented online help for all functions implemented in iterations 6 and 7. Prepare training materials and conduct training for sales, marketing, and management personnel. Conduct usability tests with a large number of actual or simulated customers. Make any needed changes to user interfaces, including online help. Update the production database with any structural changes in the test database. |
| 9 | Continue training for sales, marketing, and management personnel. Conduct performance and stress testing and make any needed changes. Conduct use acceptance testing. Place version 2.0 into production. |

## Summary

- Implementation and deployment are complex processes because they consist of so many interdependent activities
- Implementation activities include program the software, unit tests, building test cases, and integrate and test components
- Deployment activities include perform system and stress tests, perform acceptance tests, convert existing data, build training materials/conduct training, configure and set up the production environment, and deploy the solution
- Testing is a key activity of implementation and deployment and includes unit tests, integration tests, usability tests, system/performance/stress tests, and acceptance tests

## Summary (2)

- A program development plan is a trade-off among available resources, available time, and the desire to detect and correct errors prior to system deployment
- Configuration and change management activities track changes to models and software through multiple system versions, which enables developers to test and deploy a system in stages
- Versioning also improves post deployment support by enabling developers to track problem support to specific system versions
- Source code control systems enable development teams to coordinate their work

## Summary (3)

- Three options for deployment include direct deployment, parallel deployment and phased deployment
- Direct deployment is riskier but less expensive. Parallel deployment is less risky but more expensive
- For moderate to large projects, a phase deployment approach makes sense to get key parts of the system operational earlier