Murdoch
UNIVERSITY

System implementation and deployment

Topic 10

ICT284 Systems Analysis and Design

About this topic

Previous topics have covered the activities involved in systems analysis and design. In this topic, we look at activities relating to implementing the system and deploying the completed system in the organisation.

Implementation activities relate to building and testing the software and integrating all the components. *Deployment* activities involve putting the system into operation - acceptance testing by the users, training the users, converting data to the new DBMS, configuring and testing the production environment, installing the system and turning it on.

Unit learning outcomes addressed in this topic

1. Explain how information systems are used within organisations to fulfil organisational needs
2. **Describe the phases and activities typically involved in the systems development life cycle**
3. Describe the professional roles, skills and ethical issues involved in systems analysis and design work
4. Use a variety of techniques for analysing and defining business problems and opportunities and determining system requirements
5. Model system requirements using UML, including use case diagrams and descriptions, activity diagrams and domain model class diagrams
6. Explain the activities involved in systems design, including designing the system environment, application components, user interfaces, database and software
7. Represent early system design using UML, including sequence diagrams, architectural diagrams and design class diagrams
8. Describe tools and techniques for planning, managing and evaluating systems development projects
9. Describe the key features of several different systems development methodologies
10. Present systems analysis and design documentation in an appropriate, consistent and professional manner

Murdoch UNIVERSITY

Topic learning outcomes

After completing this topic you should be able to:

- Outline the activities that take place in system implementation and deployment
- Describe various types of software tests and explain how and why each is used
- Describe how to design and conduct a user acceptance test
- Briefly describe approaches to data conversion
- Briefly describe training and user support requirements for new and operational systems
- Explain in general terms the activities involved in managing the implementation, testing and deployment of a system
- Describe several approaches to system deployment and the advantages and disadvantages of each
- Describe the support activities that continue after deployment

Murdoch UNIVERSITY

Resources for this topic

READING

- Satzinger, Jackson & Burd, Chapter 14
    - There is quite a lot of detail here, but just focus on the main points.
- 6th edition: Chapter 13, Making the System Operational

Except where otherwise referenced, all images in these slides are from those provided with the textbook: Satzinger, J., Jackson, R. and Burd, S. (2016) *Systems Analysis and Design in a Changing World*, 7th edition, Course Technology, Cengage Learning: Boston. ISBN-13 9781305117204

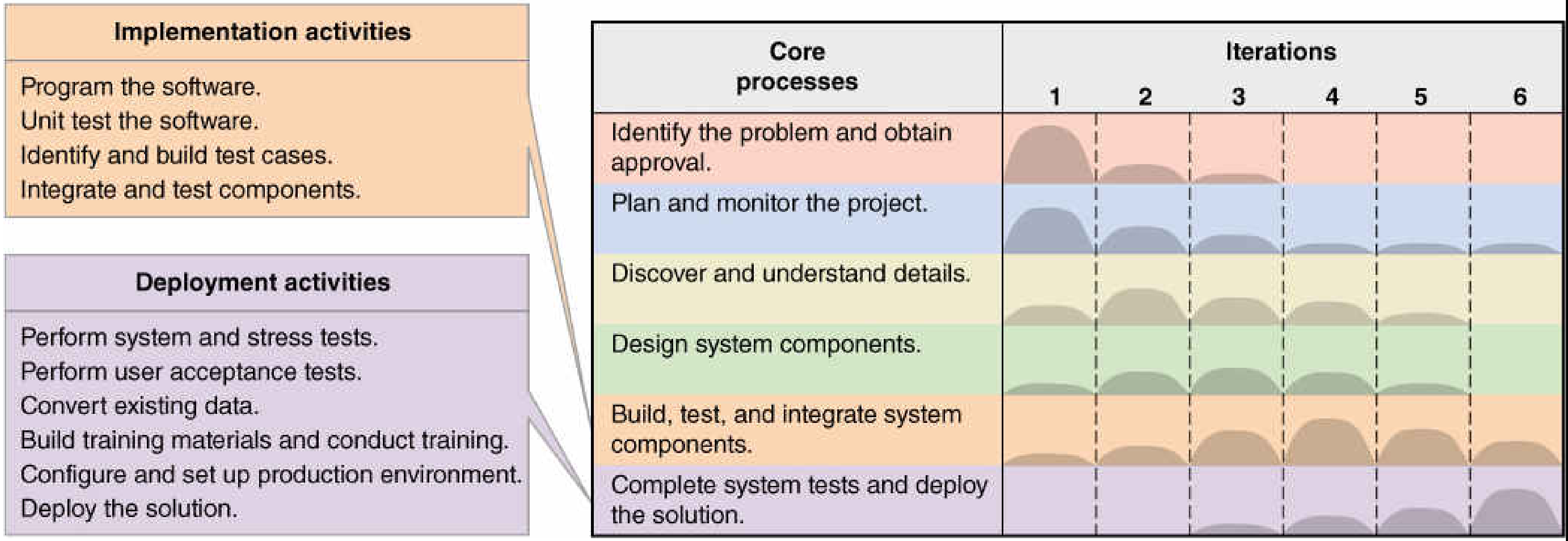
Topic outline

- Introduction
- Testing
    - Types of test
- Deployment
    - Deployment activities
    - Managing implementation, testing and deployment
    - Approaches to deployment
    - Support activities after deployment

Murdoch UNIVERSITY

Introduction

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

Implementation includes programming and testing activities. *Deployment* includes system tests, converting data, training, setting up the production environment, and deploying the solution

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Implementation and deployment

- *Implementation* activities relate to building and testing the software and integrating all the components
- *Deployment* activities involve putting the system into operation - acceptance testing by the users, training the users, converting data to the new DBMS, configuring and testing the production environment, installing the system and turning it on

Testing

Unit testing
Integration testing
System testing
User acceptance testing

Testing

- "the process of examining a component, subsystem, or system to determine its operational characteristics and whether it contains any defects"
- Testing involves defining expected operational characteristics against the specifications for functional and nonfunctional requirements – and checking actual performance against these
    - If there is a shortcoming or defect, the development team cycles back to earlier stages to remedy it
- Testing occurs during both implementation and deployment

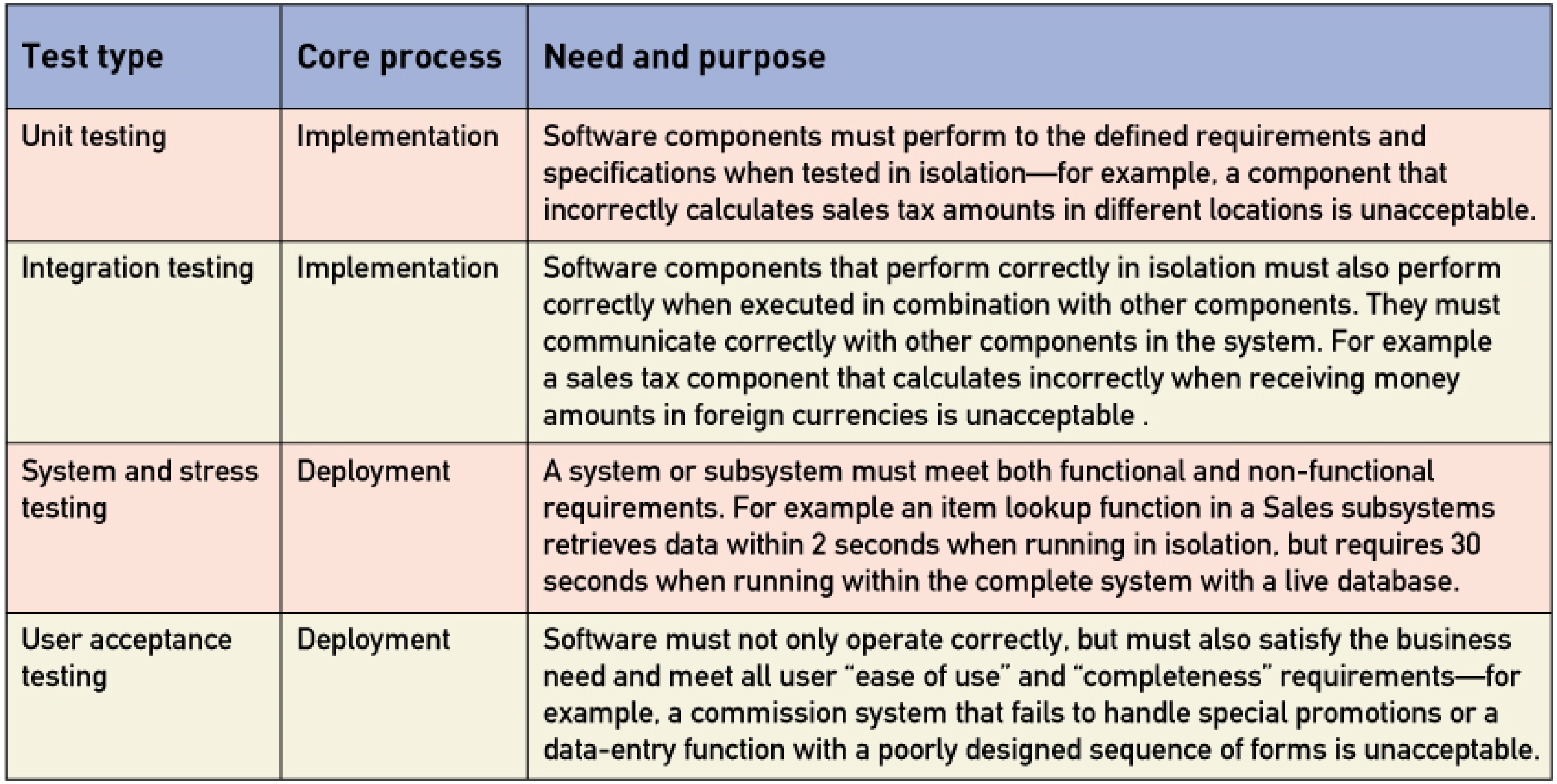
Testing

Test cases and test data must be developed:

- **Test case** – a formal description of:
    1. A starting state or condition
    2. One or more events to which the software must respond
- The test cases are represented by a set of **test data** – the set of starting states and events
- Test cases are required to full test all normal and exceptional processing situations

Common test types

Murdoch UNIVERSITY



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.

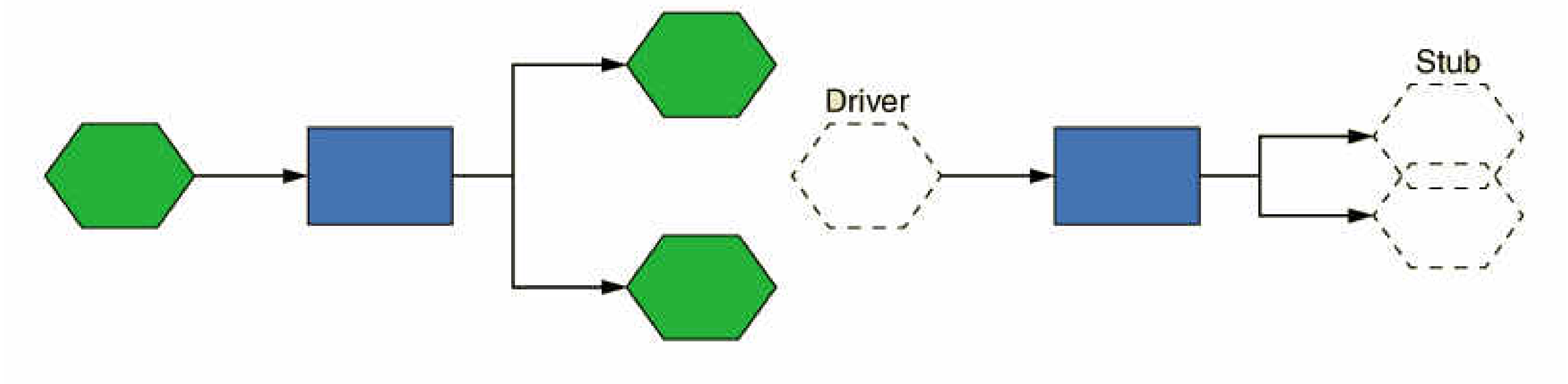
Figure 14.2 in text

Unit testing

- The lowest level and earliest testing for a software system
- Tests of an individual method, class, or component before it is integrated with other software
- Done in isolation – ensure it works correctly
- May need *driver* and *stub* methods or classes
- Done by the programmer who wrote the code – faster and simpler

Unit testing: driver and stub components

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

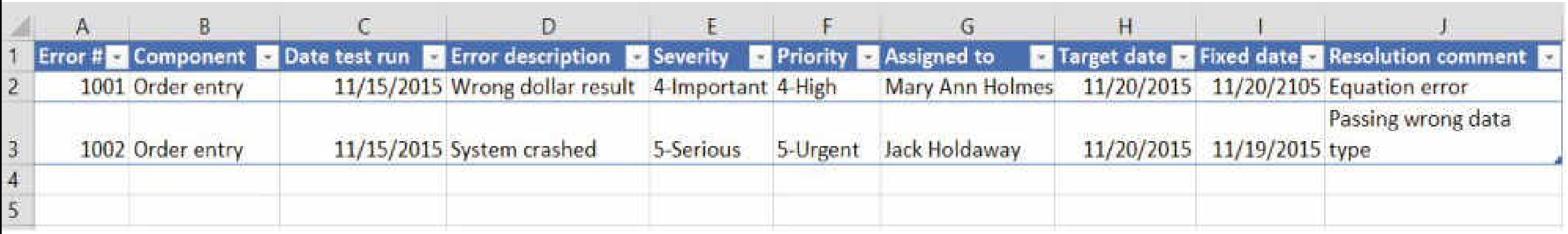
Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Integration testing

- Integration test – tests of the behavior of a group of methods, classes, or components
- After small units are tested, they are combined into a larger component and tested together
- The objective is to test the interfaces between the components, and the functionality of the entire piece of software
- Integration testing often starts small, and grows as more components are added - increasing complexity of testing

Integration testing - process

- Build and unit test the components to be integrated
- Create test data – comprehensive test data, must be coordinated between developers
- Conduct the integration test – Assign resources and responsibilities. Plan frequency and procedures
- Evaluate the test results – Identify valid and invalid responses
- Log the test results – Log valid test runs. Also log errors
- Correct the code and retest



Error #	Component	Date test run	Error description	Severity	Priority	Assigned to	Target date	Fixed date	Resolution comment
1001	Order entry	11/15/2015	Wrong dollar result	4-Important	4-High	Mary Ann Holmes	11/20/2015	11/20/2105	Equation error
1002	Order entry	11/15/2015	System crashed	5-Serious	5-Urgent	Jack Holdaway	11/20/2015	11/19/2015	Passing wrong data type

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

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, 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
  - Objects can retain internal variable values

System and stress testing

- **System test** – an integration test of an entire system or independent subsystem
- **Stress (performance) test** determines if the system can meet performance criteria such as response time and throughput
- Test the functional and nonfunctional aspects of the new system
- Can be performed at the end of each iteration, or more frequently

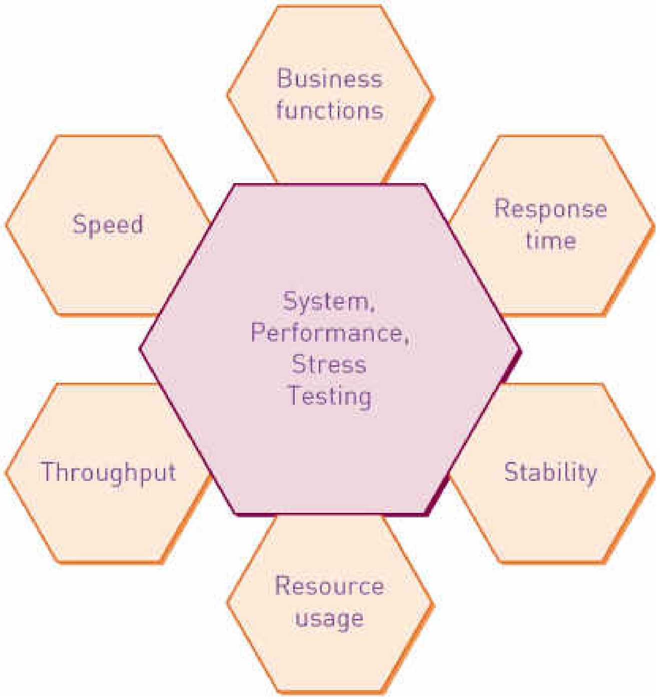


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

System and stress testing

Build and smoke test – a system test that is performed daily or several times a week

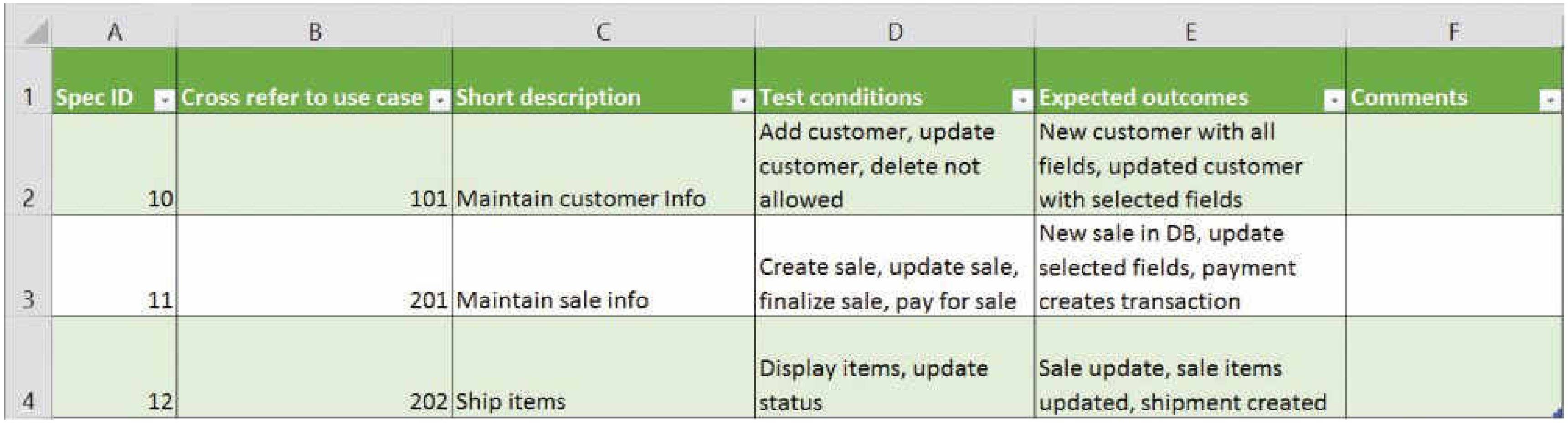
- The system is completely compiled and linked (built), and a battery of tests is executed to see whether anything malfunctions in an obvious way (“smokes”)
- Rapid feedback on integration problems as catches any problems that have come up since the last system test

User Acceptance Testing (UAT)

- User acceptance test – a system test performed to determine whether the system fulfills user requirements and can support all business and user scenarios
- May be performed near the end of the project (or at end of later project iterations)
- Often a formal activity that must be signed off by the client
- Vital part of process – if UAT not done properly, very likely the deployed system will have problems

User Acceptance Testing (UAT)

- Planning the UAT should commence early in the project and continue throughout
- Base around business events, user stories, use cases, FURPS+
- Develop test cases

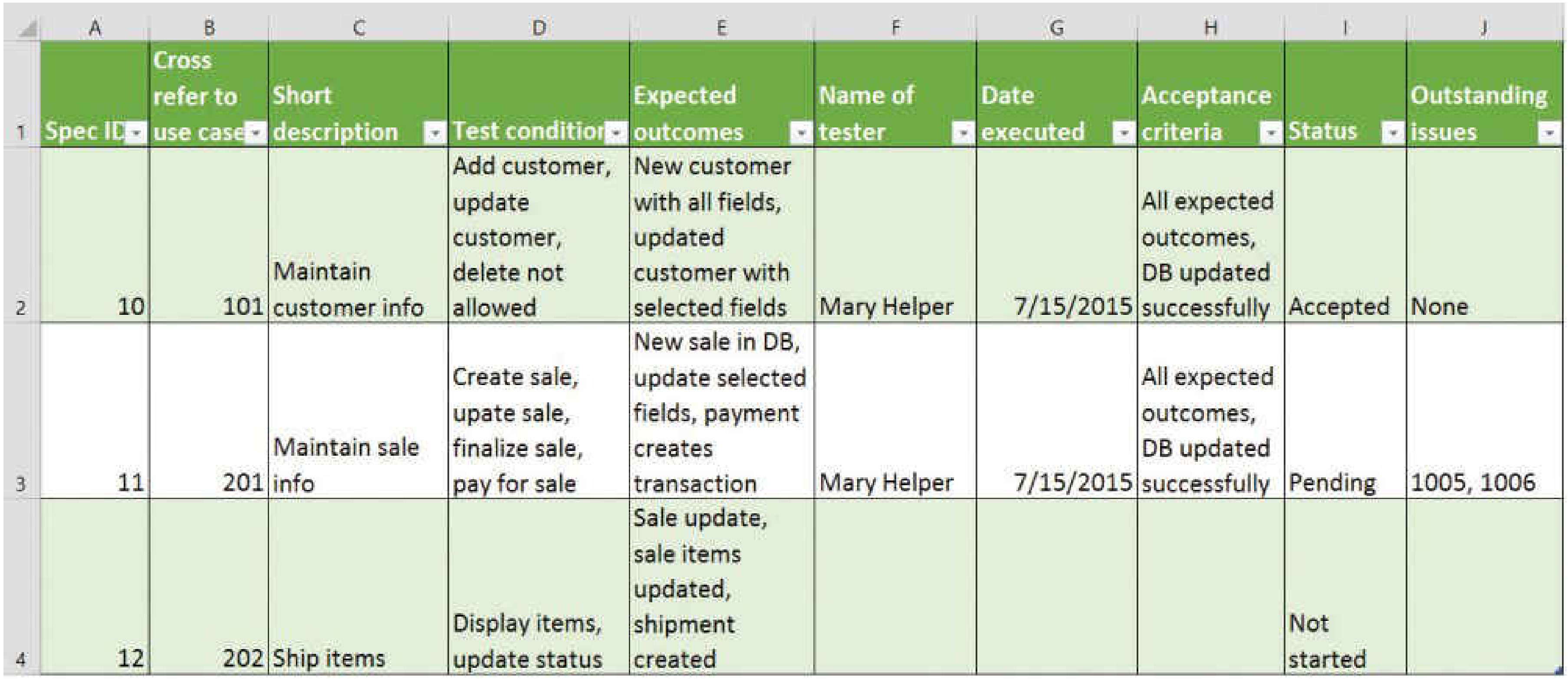


	A	B	C	D	E	F
1	Spec ID	Cross refer to use case	Short description	Test conditions	Expected outcomes	Comments
2	10	101	Maintain customer Info	Add customer, update customer, delete not allowed	New customer with all fields, updated customer with selected fields	
3	11	201	Maintain sale info	Create sale, update sale, finalize sale, pay for sale	New sale in DB, update selected fields, payment creates transaction	
4	12	202	Ship items	Display items, update status	Sale update, sale items updated, shipment created	

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

User Acceptance Testing (UAT)

- Log and track testing results



	A	B	C	D	E	F	G	H	I	J
1	Spec ID	Cross refer to use case	Short description	Test condition	Expected outcomes	Name of tester	Date executed	Acceptance criteria	Status	Outstanding issues
2	10	101	Maintain customer info	Add customer, update customer, delete not allowed	New customer with all fields, updated customer with selected fields	Mary Helper	7/15/2015	All expected outcomes, DB updated successfully	Accepted	None
3	11	201	Maintain sale info	Create sale, upate sale, finalize sale, pay for sale	New sale in DB, update selected fields, payment creates transaction	Mary Helper	7/15/2015	All expected outcomes, DB updated successfully	Pending	1005, 1006
4	12	202	Ship items	Display items, update status	Sale update, sale items updated, shipment created				Not started	

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Summing up…

Testing continues throughout the implementation and deployment core processes:

Implementation:

- **Unit testing** - Tests of an individual method, class, or component before it is integrated with other software
- **Integration testing** - tests of the behavior of a *group* of methods, classes, or components. Gradually builds up in complexity

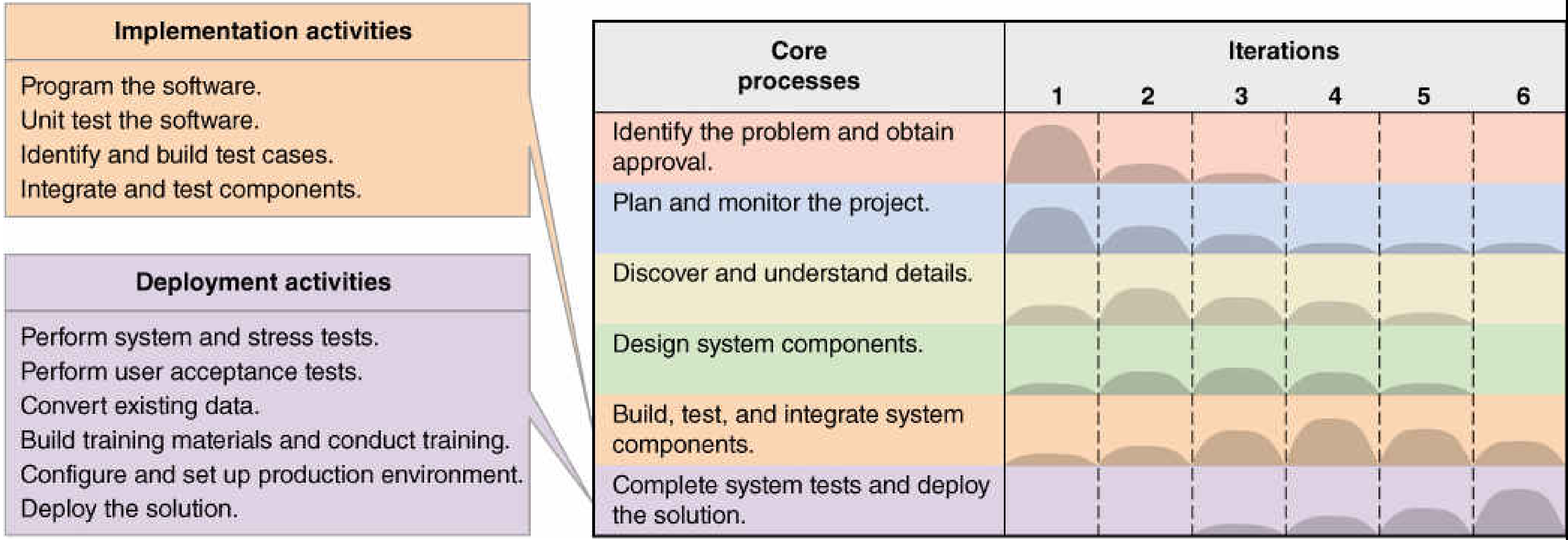
Deployment:

- **System** and **stress testing** – system testing is an integration test of an entire system or independent subsystem, while stress testing determines if the system can meet performance criteria such as response time and throughput
- **User acceptance testing** - whether the system fulfills all user requirements

Deployment activities

Data conversion
User training and documentation
Set up the production environment

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

Converting and initialising data

- An operational system requires a fully populated database to support ongoing processing
- Data needed at system startup can be obtained from:
  - Files or databases of a system being replaced
  - Manual records
  - Files or databases from other systems in the organisation
  - User feedback during normal system operation
- Can reuse existing databases, or reload into a new one

Approaches to converting to new database

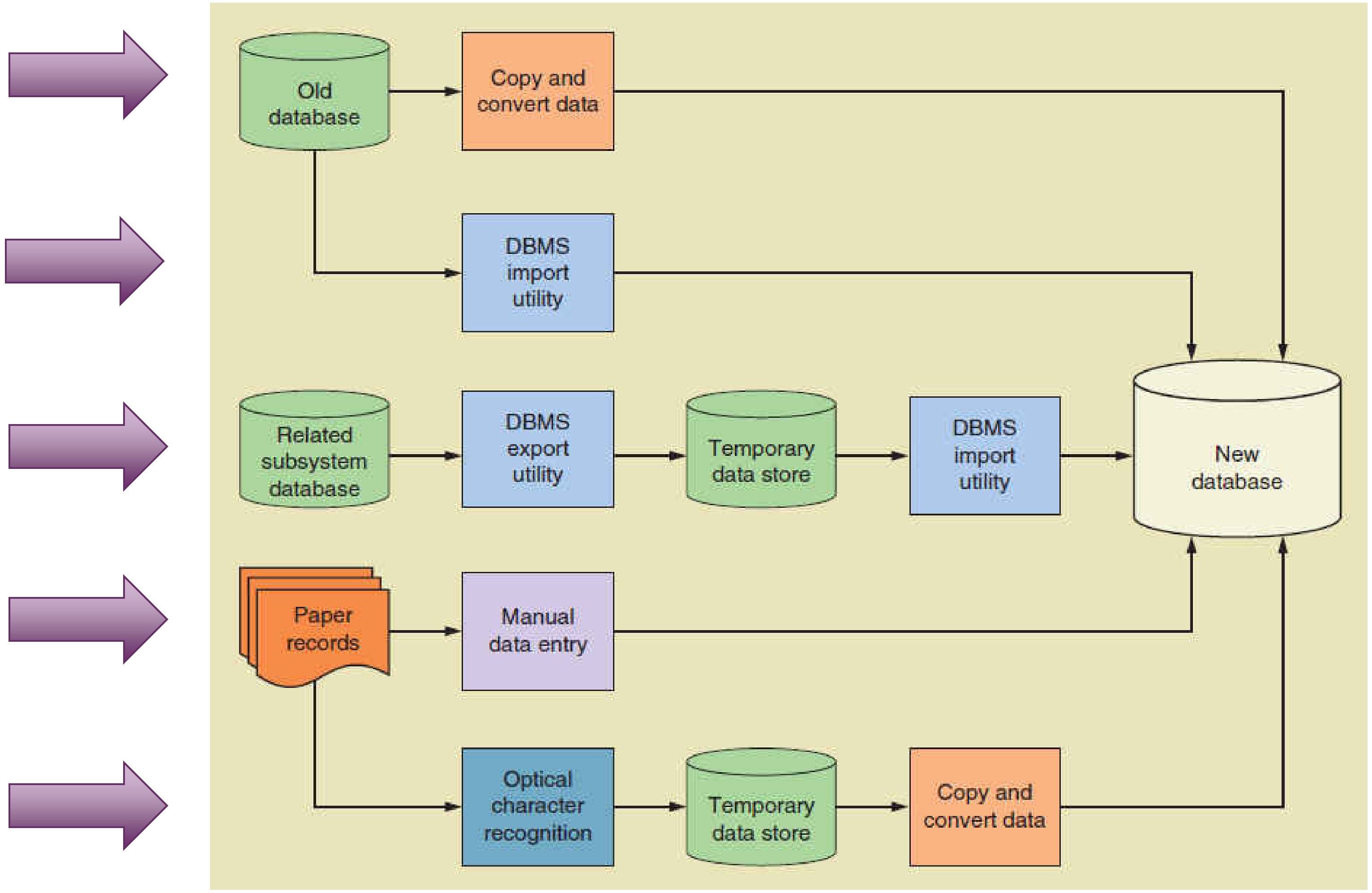
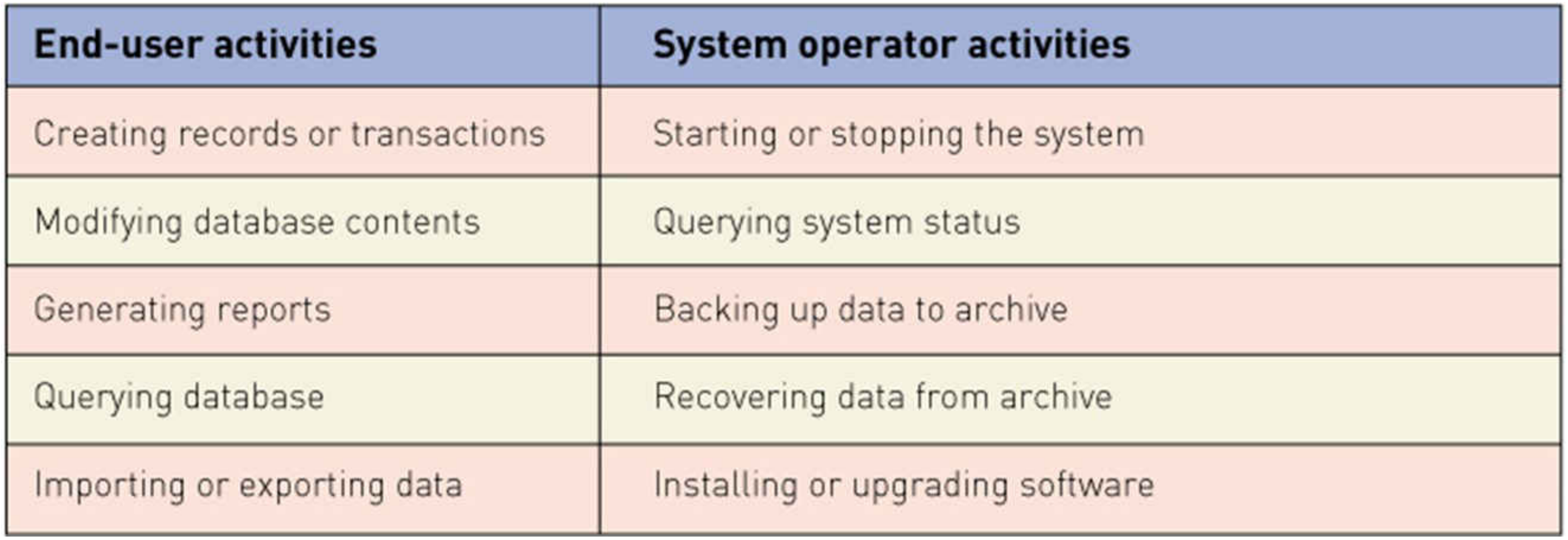


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Training users

- Training for **end users** emphasises hands-on use for specific business processes or functions, such as order entry, inventory control, etc
- **System operator** training can be much less formal, or 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

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

System and user documentation

Both system and end users require *documentation*

- **System documentation** is required for building, maintaining and upgrading the system
- Generated throughout the SDLC
- Integrated development environments ensure that system documentation is always in synch with deployed system
- **User documentation** provides support for the end users – routine operations, troubleshooting, etc
- Usually online as part of the application

Configure and set up production environment

- Applications built from software components based on interaction standards such as CORBA, SOAP, .NET etc must be configured so that they work together
- All the tasks involved in acquiring, installing and configuring the hardware and software infrastructure before the application software can be installed and tested
- Some of this will already exist, supporting existing information systems

Summing up…

- The main activities to be carried out in the Deployment phase before actual deployment are:
  - Populating or converting **databases**
  - Carrying out user training and developing user and system **documentation**
  - Acquiring, installing and configuring the hardware and software infrastructure for the **production environment**

Murdoch UNIVERSITY

Managing implementation, testing and deployment

Murdoch UNIVERSITY

Managing implementation, testing and deployment

- In a complex project there are many interdependencies that must be considered when developing a project plan – particularly in an iterative project where the system is developed incrementally

For example:

- Determining the *order* in which software components will be built/bought, tested and deployed
- Managing source code versions - e.g. with a *source code control system* that tracks and controls changes by multiple users
- Determining how the new system is to be *deployed*

Development order

- **Input, process, output**
  – based on data flow from input to output
  - simplifies testing, as input modules done first
- **Top down**
  – analyse method dependencies
  - Advantage is there is always a working version as higher level modules can call stubs
- **Bottom up**
  - do low level modules first
- **Use case driven**
  - focus on use cases first so can consider factors such as risk, user feedback, resource availability, early deployment of some parts, etc

Approaches to deployment

Direct deployment
Parallel deployment
Phased deployment

Packaging, installing, and deploying components

Approaches:

- **Direct** deployment
- **Parallel** deployment
- **Phased** deployment

Issues to consider:

- 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 familiarising customers with new procedures

- There is a trade-off between cost, complexity and risk

Direct deployment

- Installs a new system, quickly makes it operational, and immediately turns off any overlapping systems

  Higher risk, lower cost

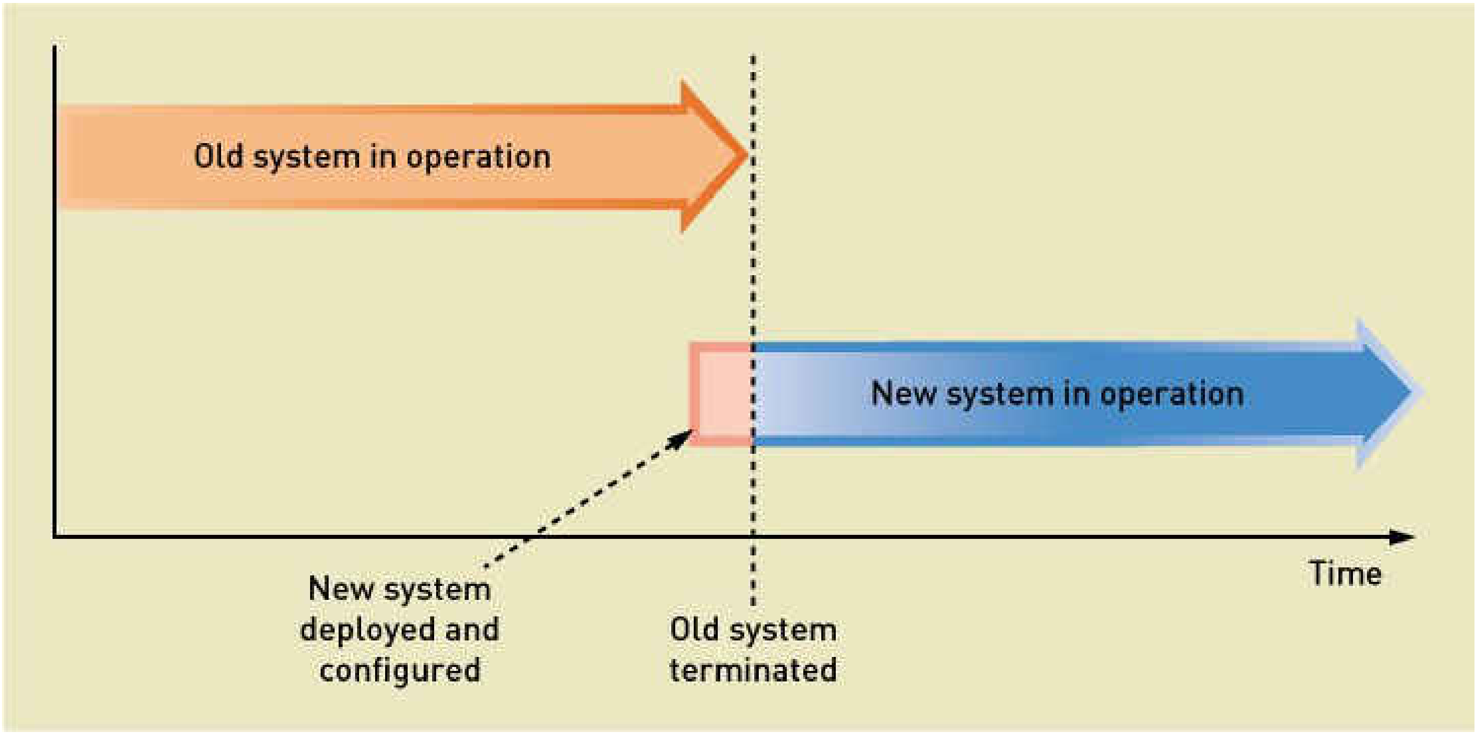


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Parallel deployment

- Operates the old and the new systems for an extended time period

  Lower risk, higher cost

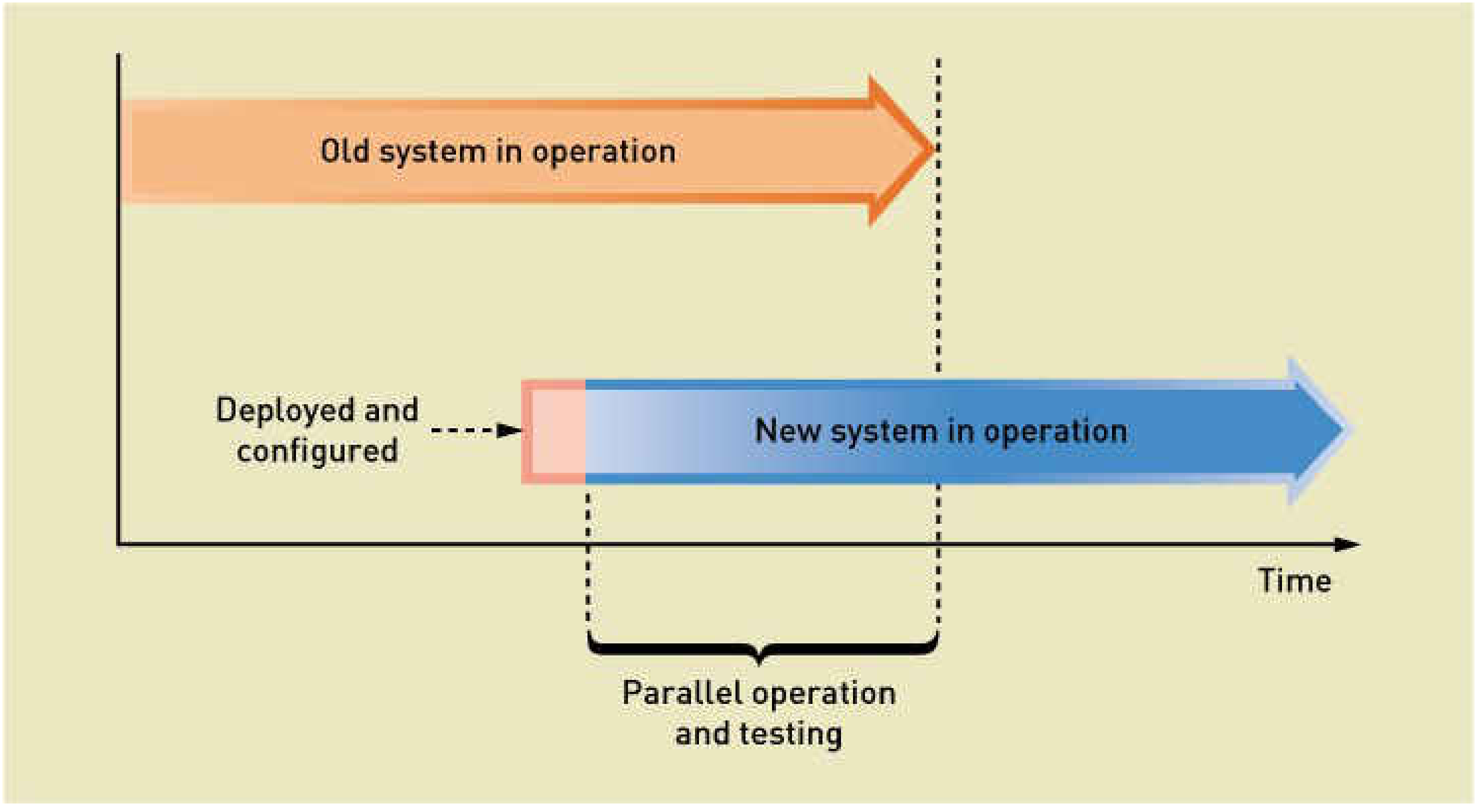


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Phased deployment

Installs a new system and makes it operational in a series of steps or phases

Reduced risk but more complexity

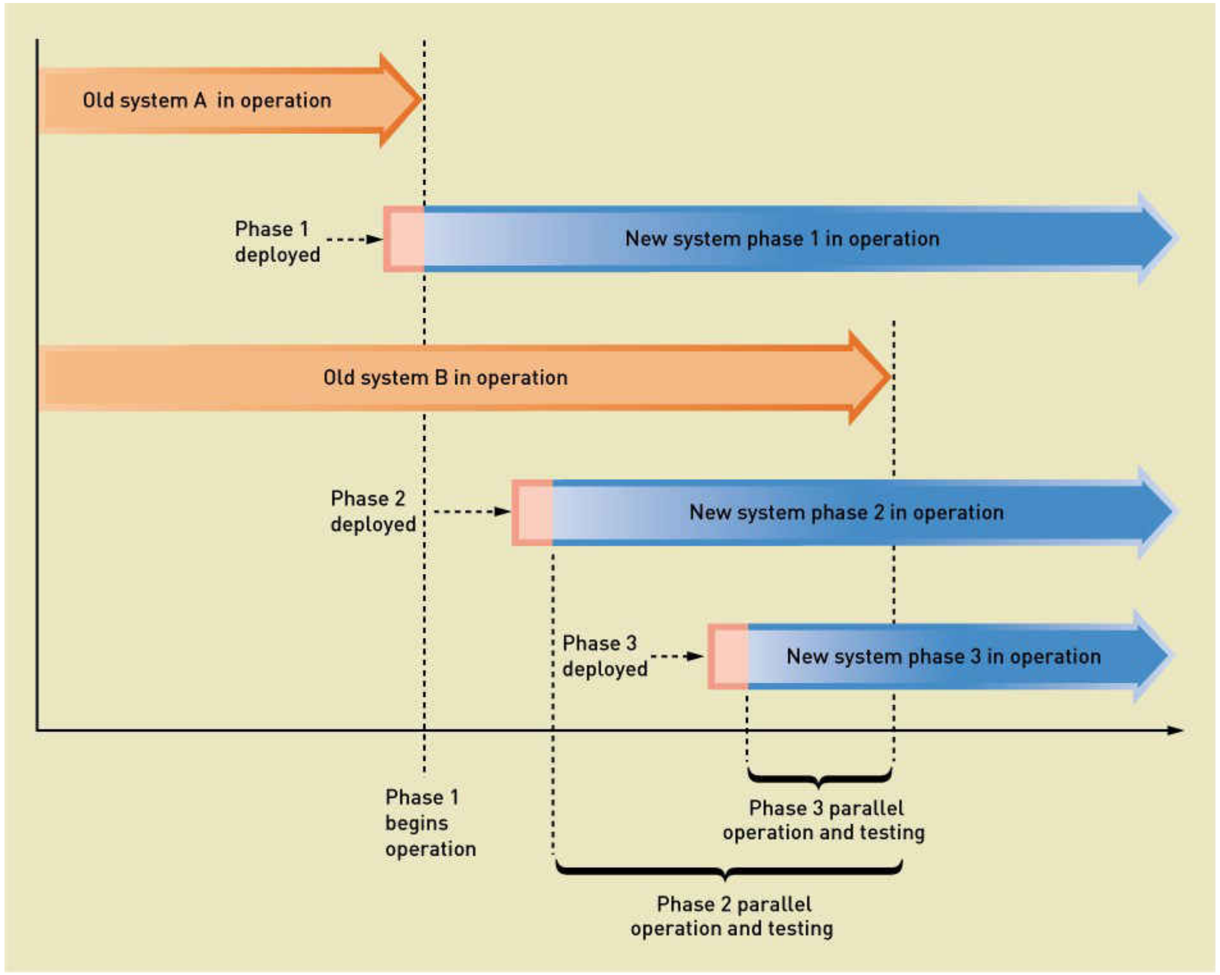

Old system A in operation
Phase 1 deployed
New system phase 1 in operation
Old system B in operation
Phase 2 deployed
New system phase 2 in operation
Phase 3 deployed
New system phase 3 in operation
Phase 1 begins operation
Phase 3 parallel operation and testing
Phase 2 parallel operation and testing

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Summing up…

- There are various ways to deploy the new system, each involving a trade-off between cost, complexity and risk:

Direct –

- switch from old to new with minimal concurrent operation.
- Higher risk, lower cost

Parallel –

- run old and new systems together for a period of time
- Lower risk, higher cost

Phased –

- the new system is introduced in a series of steps or phases
- Reduced risk but more complexity

Murdoch UNIVERSITY

Support activities

Supporting the system

- The objective of **support** is to keep the system running successfully throughout its productive life
- Predictive SDLCs usually included ‘Support’ as a separate phase after deployment
- Adaptive and iterative SDLCs tend not to, and instead may consider support to be a separate project in its own right
- Activities: Maintaining the system

  Enhancing the system

  Supporting the users

Change and version control

- Change occurs constantly throughout development and implementation, and continues (more slowly) after the system is deployed
- Managing change is essential, and change and version control tools and processes are incorporated into implementation activities and continue through the life of a system
- Complex systems are developed and installed in a series of *versions* to simplify testing, deployment and support. Multiple versions may exist, in various stages of development
- There are various version numbering schemes, but the general format is major.minor.revision, e.g. 2.3.1
-

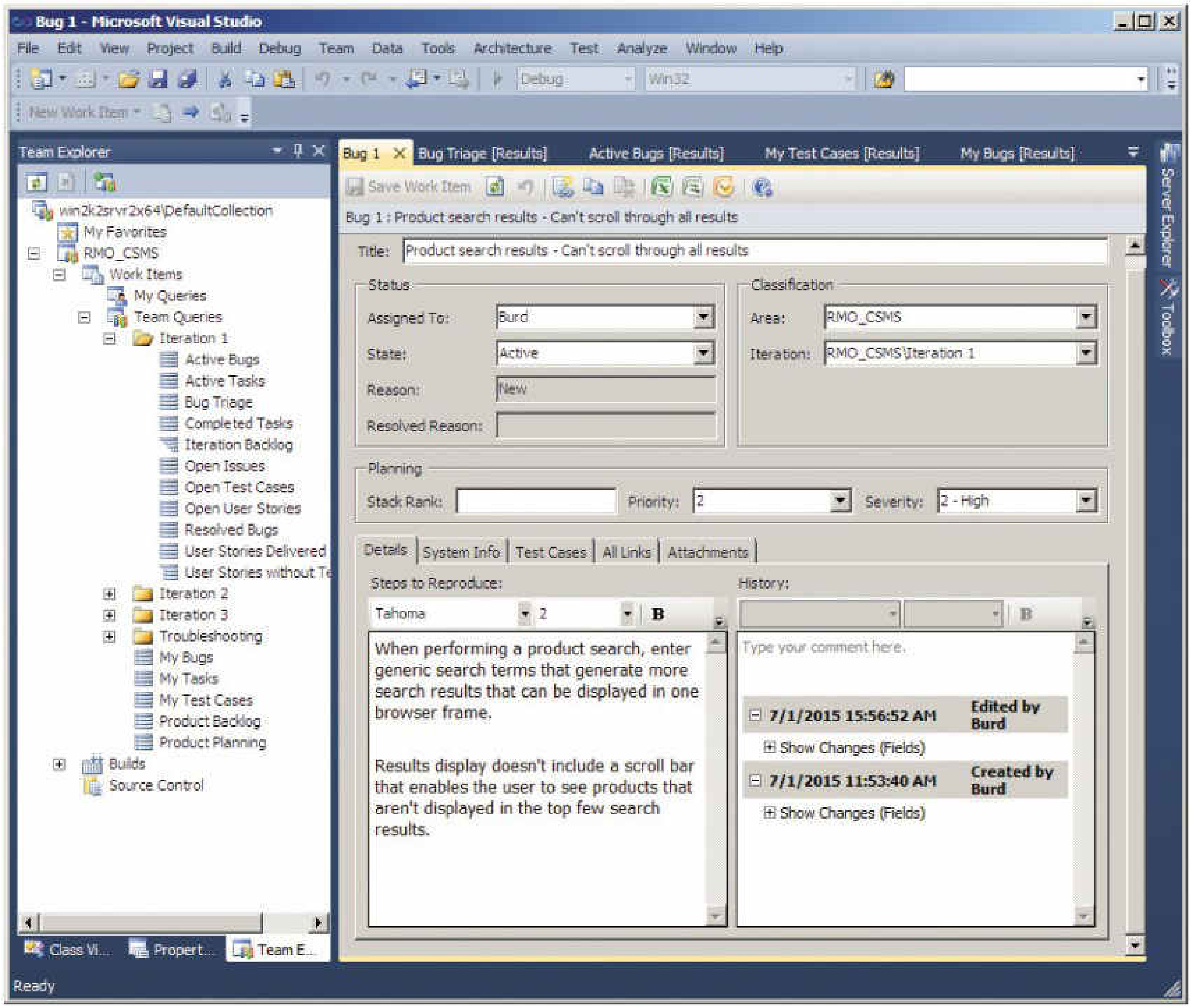
Versions

- **Test** – internal, created during development
- **Alpha** – incomplete but ready for some level of integration or usability testing
- **Beta** – stable enough to be tested by end users over some period of time
- **Production** (or **release**) – formally released to users intended to be operational for long term use
- **Maintenance** – a system update to a production version to provide bug fixes and minor updates
- All beta and production versions must be stored as long as they are installed on any user machines

Error reports and change requests

- Any change requests or bug reports are handled through formal control procedures to ensure changes are adequately described and planned
- There is usually a formal reporting method so that all reports can be managed centrally and any impacts evaluated
- Approved changed are added to a list for subsequent, budgeting, scheduling, planning and implementation
- Where possible changes are implemented and tested on a copy of the production system, and after successful testing the copy becomes the new operational system

Example: Error report in Microsoft Visual Studio



Bug 1 : Product search results - Can't scroll through all results

Title: Product search results - Can't scroll through all results

Status
Assigned To: Burd
State: Active
Reason: New
Resolved Reason:

Classification
Area: RMO_CSMS
Iteration: RMO_CSMS\Iteration 1

Planning
Stack Rank:
Priority: 2
Severity: 2 - High

Steps to Reproduce:

When performing a product search, enter generic search terms that generate more search results that can be displayed in one browser frame.

Results display doesn't include a scroll bar that enables the user to see products that aren't displayed in the top few search results.

History:

Type your comment here.

7/1/2015 15:56:52 AM Edited by Burd
Show Changes (Fields)
7/1/2015 11:53:40 AM Created by Burd
Show Changes (Fields)

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Summing up…

- Once the system is deployed, it must still be supported so that it continues to operate productively
- Supporting the system may be considered a phase of the SDLC, or in an iterative project, a project in itself.
- Support activities involve maintaining and enhancing the system, and ensuring users continue to be supported, including requesting fixes or changes

Topic learning outcomes revisited

After completing this topic you should be able to:

- Outline the activities that take place in system implementation and deployment
- Describe various types of software tests and explain how and why each is used
- Describe how to design and conduct a user acceptance test
- Briefly describe approaches to data conversion
- Briefly describe training and user support requirements for new and operational systems
- Explain in general terms the activities involved in managing the implementation, testing and deployment of a system
- Describe several approaches to system deployment and the advantages and disadvantages of each
- Describe the support activities that continue after deployment

Murdoch UNIVERSITY

What’s next?

We’ve now covered all the activities in the SDLC and some of the tools and techniques involved. In the final two topics, we change the focus to managing the systems development process itself. This involves consideration of project management activities and the choice of development methodology.