SE202
Introduction to Software Engineering

Lecture 8
Software Deployment

Today

- Software Deployment
- Deployment plan
- Cutover
- Deployment task
- Deployment mistake

Scenario

- Suppose you have a plan to deploy a new application on customer's new computers. The new computers arrive on schedule and they work just fine, but there's something wrong with the network and users are not getting the bandwidth they should so the users can only process 4 or 5 jobs per day instead of the normal 15 to 20.

What should you do?

Move the users onto the new system right away?

OR Wait until there's a fix?

Software Deployment?

- Deployment term: *installation and release*
- The process of putting (installing) the finished application in the target environment (the users' hands)
  - Consider the scope of the deployment, make a plan and follow the plan
  - If big unexpected problems occur, roll back any changes you've made and try again later.
  - Use system conversion strategies to make things easier.
- You need to know when to abandon a deployment attempt and try again another day!!

Deployment scope

- Number of user
- Size of application
  - Data involved
  - Number of external systems
  - Quantity of codes

Deployment plan

- List the successful steps and detail how each step is supposed to work
- Next, for every step, list the way that step could fail and describe work-arounds or alternative approaches
- Next, write a rollback plan that lets you undo everything you've done
  - For a major restoration possible → do complete images of the computers you're updating
  - The point of no return is where it would be more painful to roll back a failing deployment than to press ahead. (If you have a good rollback strategy, then you don't need a point of no return.)

System Conversion

- System Conversion is the process of moving users to the new application.

- 6 ways of System Conversion
  - Staged deployment
  - Direct conversion
  - Parallel conversion
  - Incremental Deployment
  - Pilot conversion
  - Gradual Cutover

Staged Deployment

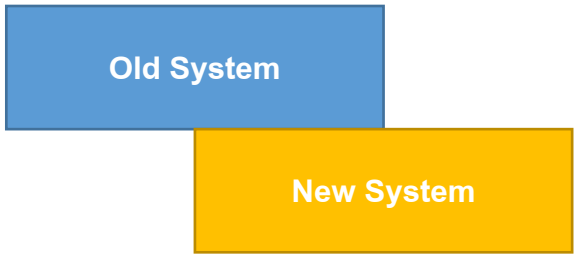
- Testing the new application in a staging area to find and fix a few final bugs before delivering on the users.
  - Staging area is an environment that’s more realistic than the one used by the developers,
- Engaging of power users in this deployment is very useful.

Direct conversion

- This conversion is done simply by abandoning the old and starting up the new system, the old system is no longer available to fall back on.
- The direct approach is not recommended precisely because it is so risky.

- Pro?
- Con?

Parallel conversion

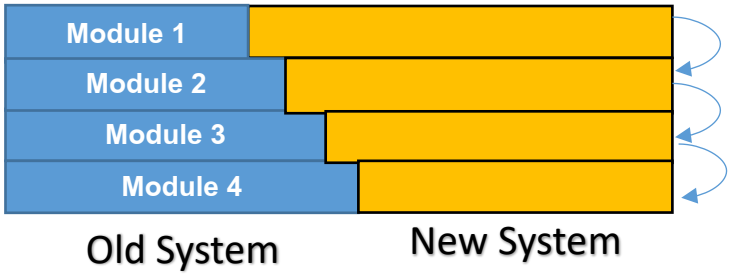


- The old and new systems are operated side by side until the new one proves to be reliable.
- This approach is low-risk. If the new system fails, the organization can just switch to the old system to keep going. However, keeping enough equipment and people active to manage two systems at the same time can be very expensive. Thus, the parallel approach is used only in cases in which the cost of failure or of interrupted operation is great.
- Old and new systems are used at same time.

Parallel Testing

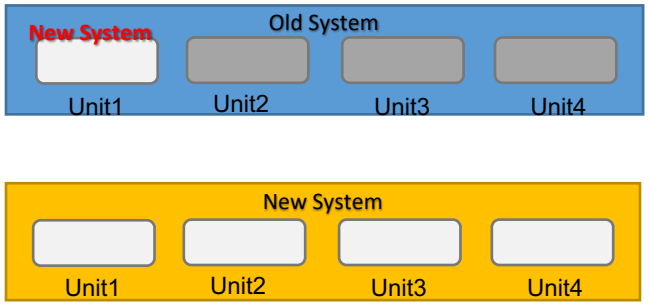
- Simply run old system and new system in parallel
- Fully validate and sign-off the platform and safe in the knowledge that it completely meets their needs.
- Start migrating the users to the new system when you enough confidence in the new system
- The downside is cost, maintaining two environments is clearly more expensive than maintaining one.

Incremental Deployment



- Releasing the new system's features to the users gradually.
- First, you install one feature (possibly using staged deployment or gradual cutover to ease the pain). After the users are used to the new tool, you give them the next feature.
- This method doesn't work well with large monolithic applications because you usually can't install just part of such a system.

Pilot conversion



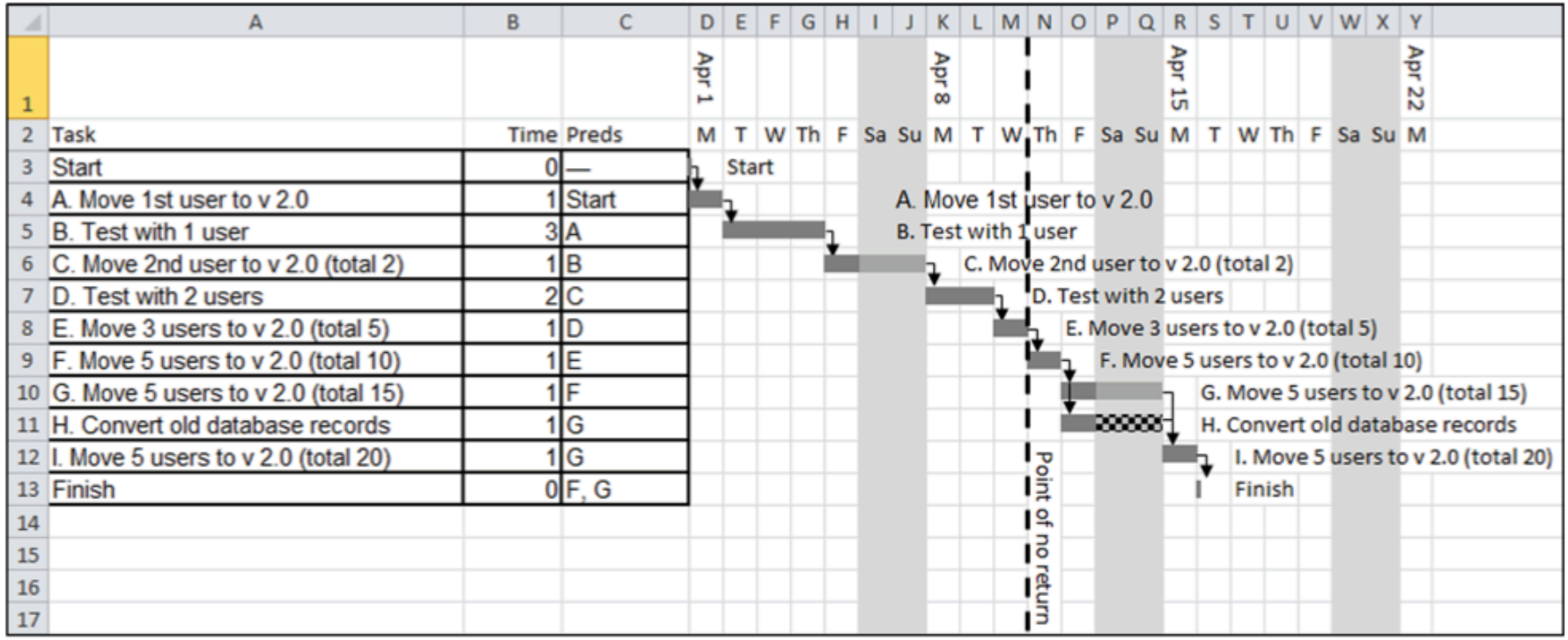
- In the pilot approach, the new system is tried out in only one part of the organization. Once the system is working smoothly in that part, it is implemented throughout the rests of the organization.
- The pilot approach is certainly less expensive than the parallel approach. It also is somewhat riskier. However, the risks can be controlled because problems will be confined only to certain areas of the organization. Difficulties will not affect the entire organization.

Gradual Cutover

- You Install the new application for some users while other users continue working with their existing system.
- You move one user to the new application and thoroughly test it.
- When you’re sure everything is working well, you move a second user to the new system. When that user is up and running, you install a third user, then a fourth, and so on until everyone is running the new application.

Example of AdventureTrek 2.0 deployment plan (Gradual cutover)

Task	Time	Preds
Start	0	—
A. Move 1st user to v 2.0	1	Start
B. Test with 1 user	3	A
C. Move 2nd user to v 2.0 (total 2)	1	B
D. Test with 2 users	2	C
E. Move 3 users to v 2.0 (total 5)	1	D
F. Move 5 users to v 2.0 (total 10)	1	E
G. Move 5 users to v 2.0 (total 15)	1	F
H. Convert old database records	1	G
I. Move 5 users to v 2.0 (total 20)	1	G
Finish	0	F, G



This schedule takes 11 work days to migrate all 20 users to AdventureTrek 2.0

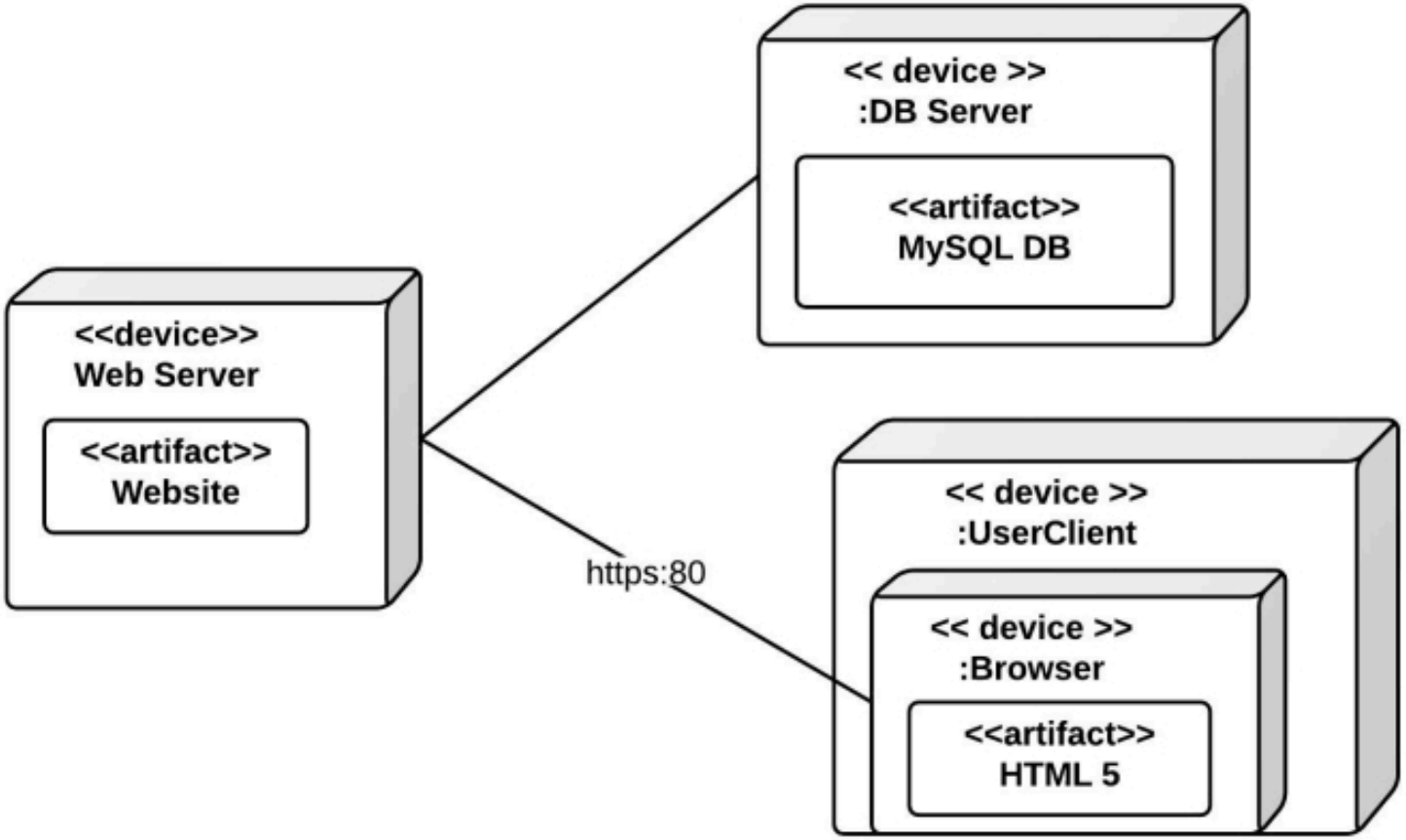
Deployment tasks for a large deployment

- Physical environment
- Hardware
- Documentation
- Training
- Database
- Other people’s software
- Your software

Deployment diagram

- Static deployment view of a system
- Physical allocation of components to computational units
- Node – computational unit
- Edge - communication

Example of deployment diagram



https://www.lucidchart.com/pages/uml/deployment-diagram

Deployment mistakes

- Assume everything will work
- Have no rollback plan
- Allow insufficient time
- Don’t know when to surrender
- Skip staging
- Use an unstable environment

Homework assignment

- Answer the following questions:

1. Suppose you've written a small tool for your own use that catalogues your collection of CDs. You're planning your third upgrade and you need to update the database design. Which cutover strategies should you use?
2. Suppose you're writing an application that includes a lot of separate tools. One creates work orders, a second assigns jobs to employees, a third lets employees edit jobs to close them out, and so forth. Which cutover strategies could you use when deploying a new version of this application?
3. Suppose you're writing a large application with thousands of users scattered around different parts of your company. Which cutover strategies would you use?