# Product rollout/deployment

- Sooner or later your product is ready for prime time
- The process followed to have it go live will leave a lasting impression on your clients and end users
- For complex systems, there can be many parts that need to be set up properly for the whole system to work
- We need to ensure that we are ready for it to go live, that our clients are ready for it to go live, and we have contingency plans for things that may go wrong

# Smaller, client-side installations

- Even with small applications, many installation and configuration concerns to address
- What access/permissions are necessary
- Does it alter system settings (paths, keys, startup, ...)
- Which option(s) are user-configurable, to what degree
- How much support do we provide for aborting installation
- Do we support multiple installations

# Uninstalls, re-installs

- If user uninstalls, do we restore system settings to what they were before we installed?  Will that impact changes user has made since then?
- If user re-installs, but had data and custom settings from previous install, how much of that do we retain, and how much do we overwrite?

# Patches and upgrades

- How do we notify user and let/get them to patch/upgrade?
- How does patch/upgrade handle existing user data and customized settings?
- Does patch/upgrade work by completely replacing altered files, or by editing them?
- What permissions are necessary for patch/upgrade to run?
- Does patched system give same behaviour as clean install of latest version of system?

# Push vs pull

- For systems that will be used by many clients, do we push the new system out to everyone, and make them upgrade, or do we let them choose if/when we want to upgrade
- The forced upgrade can irritate some users, especially if there are bugs/glitches/changes from the ‘old way’
- Giving them the choice on if/when tends to increase the number of systems we wind up actively supporting
- How do the installations know there is a new version waiting?  Do they check in/call home periodically to see?

# Larger deployment strategies

- If it's a huge system, are we deploying all the parts of it at once, or in stages?  There can be more to manage in the "all at once" approach, but if we do it in parts then the old parts and new parts need to be compatible
- If it's being installed in many sites (e.g. a chain of warehouses), do we do a couple of test sites first?  Again, it reduces the scope of the test installation, but means the new system at the test site may need to be able to communicate with the old system at other sites

# Getting the client ready

- A new system may represent a substantial change in the client's business processes (maybe we're rolling out a new system for the tellers at a bank, or new interface for the agents at an airline, or new software for the cash registers at a grocery store)
- The users of the new system may need training on it
- Their technical/support team needs to be trained/prepped for it
- Their customers might need to know that hiccups/slowdowns may occur during the changeover

# Getting the hw/sw ready

- A new system might require new hardware, or reconfiguration of old hardware – this could involve devices at the user end, various servers, gateways, firewalls etc
- The new system might require installation or upgrades to a range of software across the various systems
- The new system might involve changes to a variety of configuration settings, files, permissions
- The new system might require connections/accounts for a variety of different services
- The new system might require importing or converting a variety of data files/data bases

# Documenting and automating

- The changeover can be a very complex process
- Solid documentation/guides should be created for each portion of the process, and reviewed by the people involved
- Where possible, scripts and tools should be used to automate the process, since manually carrying it out is likely to be error prone

# Testing the process

- Ideally, we want to trust that our deployment strategies will actually work
- This requires testing as much of the process as possible, before doing it “for real”
- This, in turn, requires setting up a test environment that is as close as possible to the actual working environment, and testing and refining our processes there

# Contingency planning

- What if something goes wrong?
- We’d like contingency plans for predictable issues (a particular part of the process fails), reflecting the impact they will have on the client/users
- Can we roll back to the old system?  Or part of it?
- Which parts can we reasonably provide workarounds for?
- Have the clients/users been adequately warned about and prepared for the possibility of significant delays?