Software-intensive systems

- Real systems have software and hardware and people and...
- During development, not such a big deal
- At deployment time, things get interesting
Types of system

- White box app
- Commercially deployed app
- In-house app
- Infrastructure SW / tool / library
- Embedded system
- ...

The deployment life-cycle

- Final QA
- Packaging
- Distribution
- Customer validation
- Customer acceptance
Final QA

- Regression test: kill all regressions
- User test: have a typical user run the program for a while
- High-level validation: can anyone think of problems to be solved before shipping?
Known product

- You should be able to reproduce the build you are about to package **perfectly, many years from now**
  - All code in single SCMS? In archival tar/zipball?
  - Formats? **Tools**?
  - HW dependencies?
Traceability

- Can the fielded version be matched with what you are about to package?
  - Have you preserved tagged intermediate work products?
  - Is the fielded product tagged with sufficient identifying information (versioning)?
Packaging

- **What packaging options do you have for what you are doing?**
  - Easy case: it's in ROM
  - **Hard case:** it's a network-installable package for some obscure platform

- **Try to conform to pkg stds**
Standard packages

• The user, platform, etc have **standard expectations** for software packaging: follow them!
  – MacOS, Win installers
  – Linux packages
  – etc
User documentation

- Will the user know how to work this thing?
  - We should have set up a documentation plan during requirements
  - Validate docs now
- Will the user know how to read the docs?
Delivery

- Giving a white box to a distributor is easy
- Setting up network delivery is harder
  - security issues
  - payment issues
- For other things, delivery means more than this
Secure Deployment

- Viruses and trojans in package?
- Possibility of outside tampering w/ installation?
- The dreaded "laptop from outside"
- The dreaded disgruntled
**Delivery of infrastructure**

- Often accompanied by a human expert for installation
- May involve delivery of training also
- Needs to be **coordinated** with other on-site SW
- Choose an appropriate time
Product families

• Have to make sure that the right software gets to the right place
  - clearly mark everything
  - adaptable SW products are better than SW product families
Customer validation

- Maintenance costs start now
  - Even in white-box world, unhappy customers call up
  - Enterprise customers will make you get it right
- Glad you have clear, clean validated requirements?
Dodging delivery-day disasters

- Do what you can to
  - make sure there are backups
  - make sure that mission critical systems are not disrupted
- Run infrastructure in parallel
- Remember, HW is cheap. You are expensive. Customer's business is priceless
Customer acceptance: maintenance begins

- Following suggestions given here can dramatically decrease maintenance
- Still, not having customers would be easier
- Open source has a somewhat different model for all this
Open source: continuous delivery

- In open source, delivery starts as soon as there's code
  - Source first
  - Binaries second
  - Packages last
- **No penalty** for small incremental deliveries
Open source: customer validation

- Open source is incrementally customer validated, also
- Exception: “big industry dump” packages
  - These are not accepted quickly
- Techie open source users run the process backward
Evolution of software deployment

- **1970s**: have a magtape with a system product
- **1980s**: have a floppy with an end-user application
- **1990s-present**: have a package integrated into an “OS”
- **2000s**: ???