System Integration Challenges

- What changes go in what release?
- How are releases numbered/identified?
- How do we allow continued development while integrating, testing, and releasing?
- What if not all code is defect free? Can we release some defect fixes and not others?

System Integrator

- Manages build environment
- Manages branches/tags
- Manages acceptance testing/functional verification

A Release Numbering Strategy

x.y.z[[A|B] w]

x = the major release number

y = the feature release number

z = the defect repair number

A = alpha (internal release for testing)

B = beta (external release for testing)

w = the iteration of the alpha or beta release

System Integration Strategies

- "Latest is greatest" system integration
  - Most recent check-ins are correct and cause no functionality regression in the product
  - Assumes that recent check-ins don't adversely affect other files
  - Requires team to address all errors by recent check-ins
- Modularized system integration
  - Source base is divided into modules (components)
  - Teams organized around modules
  - Modules are individually developed and tested
  - System integrator manages branch where module changes come together

Modules

- Interface
  - Functional verification focuses here
  - System integration cares most about interface
- Internals
  - Responsibility of module team

System Integration Process

- Scheduled vs. Event-Driven
  - Scheduled example:
    - System Integrator checks out source at noon every Thursday and builds, integrates, and tests all changes to date
  - Event-Driven example:
    - System Integrator is notified when a set of changes is complete (e.g. a patch or new version). SI then checks out source, builds, integrates, and tests.
- Parallelism
  - Can we support parallel/staggered build/test/release of different change sets through branches and multiple system integrators?

Resolving System Integration Problems

- Unstructured
  - Highly competent integrator
    - Knows development staff to find help
  - Hard to train and retain such staff
- Structured
  - Hierarchical problem escalation
  - Documented protocol to contact development groups

Exercise

- If your team increased by 30 developers and you needed to complete all of the original requirements plus a few more in two months, how would you organize the architecture and team?
  - Review Architecture
    - Identify existing and potential components/modules and interfaces
  - Make system integration plan
    - Branching/tagging strategy
    - System integration schedule
    - System integration protocol
      - Criteria for checkin, integration candidate, integration complete, integration rejection, problem escalation, …