Announcements

Always be looking ahead:

OpenSource

- Demo/Findings (**next week**) – AWS **Academy confirm access today and if you are using it or demo from laptop**

Exam #1 on Tue Feb 15

- There will be a lot on the exam that comes only from the readings. Catch up now!

Project

- ERP Research & App Selection – must **procure** asap
- Stand-alone: Tie In (not expected at R1)

R1 (and check **rubric**) *Shall we negotiate?

First a word about Leadership

SWEN-343

Leadership

What is a leader?

A Leader is Not

Telling people what to do
Someone who plays the blame game

Zuckerberg spends most of this time asking questions

A Leader is

1. Helping others in time of need
2. Being prepared
3. Leading by example
4. Being the SME (Subject Matter Expert)

Companies want leaders, even if you're not the boss or you're "at the bottom"

How Can you Be Leader on Your Team(s)?

Systems Integration

SWEN-343

Project Notes - Will need APIs Between Components

- Support component **stub**bing
- Separate (sometimes **dis**parate) group concerns
  - Grading
  - Customer requirements
  - Testing/security
- Interchangeability
  - Project risks
  - Changing Requirements (**ask** the customer!)
  - Properly modeling & architecture

Agenda

Challenges to overcome
Techniques for integration

Key Questions

How to integrate?
- At the flip of a switch?
- Components/piecemeal?
- What levels?

How to break up work?
- Who is responsible

How to test?
- Was integrated correctly the 1st time?
- Root cause of discovered errors
  - They will happen

Systems Integration is the process of:

"Assembling the constituent parts of a system in a logical, cost-effective way, comprehensively checking system execution (all nominal & exceptional paths), and including a full functional check-out."

Integration Tips

- Systems integration should be done at the system level
  - Why not unit?
    - coupling
    - others…..
- Don’t make it a “backend” task
  - May actually be a primary driver for architectural/design decisions.

Some Integration Concerns

Technological
Security
Scalability
Reliability
Maintainability

Phases of Integration

- What components do you need/not need?
  - What are the nice to haves & what are the risks
- How will it map into your current design & components?
- Select appropriate systems
- Subsystem integration
- Integration, analysis, verification
  - analysis → functionality, security, performance

Systems Integration: Big Bang

Integration starts late because of late components, resulting in a flawed and immature delivery.

Integration is expected to occur instantaneously, feeding an abbreviated and incomplete Test process, in order to recover schedule and meet the delivery deadline

Focus is on components rather than system capabilities; the Integration team rarely understands the overall system concepts

Integration starts when the hardware and software are ready, so it uses the delivered hardware and software for all activities

Recommended Process

Adopt a Continuous Integration model rather than a Big Bang Integration model. Establish an Integration rhythm that is essentially independent of the development team.

Create a Systems Integration team of Responsible Engineers that knows the entire system and follows the program from **Requirements Definition** through Acceptance Testing and Operations. Design and Test engineers provide required support to REs during integration

Manage system integration and system test based upon subsystems that can be end-to-end tested against system level requirements; manage system design & development based upon components that can be independently developed and checked.

Recommended Process (con't)

Define a Configuration Management process such that the System Integration and Configuration Management Teams build and control the hardware & software configurations.

Develop component and subsystem specifications to the extent that they are needed in order to define component checkout & subsystem verification procedures.

Perform component-level checkout to satisfy Integration entry criteria.

Continuously perform regression testing; create internal and/or external automated test tools that greatly reduce the emphasis on man-in-the-loop testing.

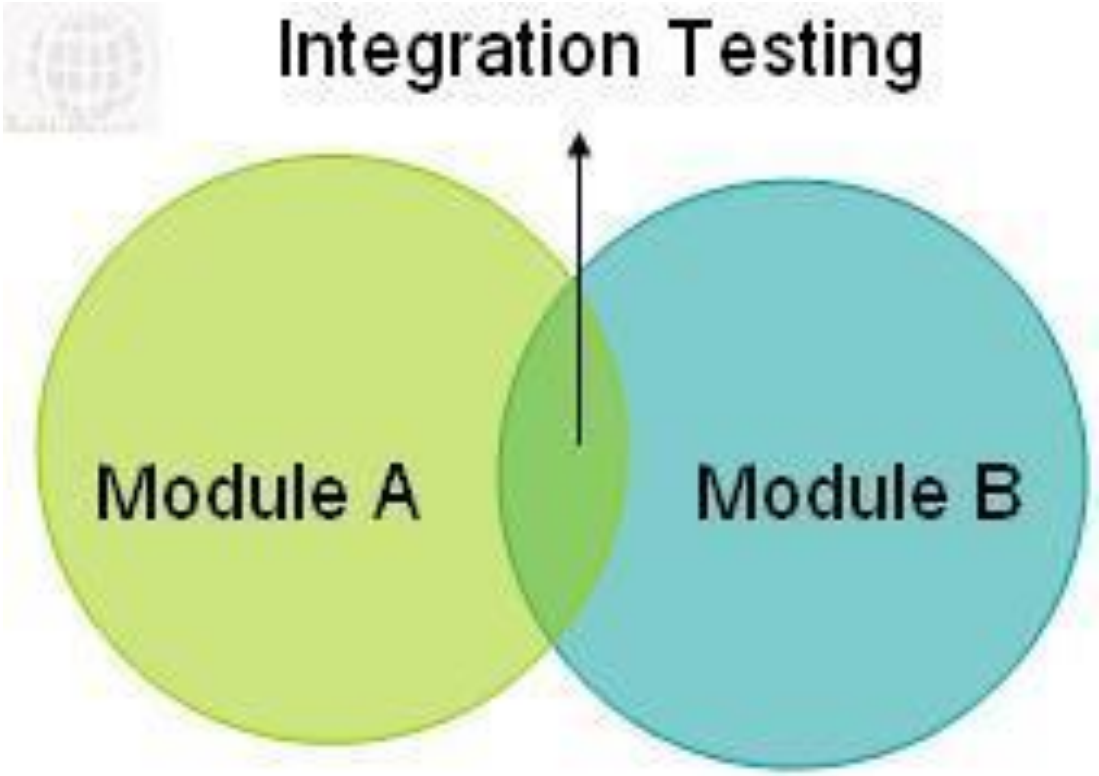
Recommended Process (con’t)

Track integration progress based upon completing subsystems that have been verified end-to-end against system-level requirements.

Augment requirements-driven testing with stress testing and long mission threads testing in order to promote a robust system.

Create a **System Architecture Skeleton (SAS)** very **early** in the program and use it as the framework for Subsystem Integration as components are added incrementally

Integration



Integration Testing

- You will be (likely) using many technologies/components
  - Maybe more than you think
- How will you ensure that all items are properly integrated?
- How will you find problems?
- How will you determine who should fix the problems?

Integration Testing Strategies

Types of Integration Testing:

- **Bottom-up:** Drivers & stubs
- **Top down:** Top layer & then sub layers
- **Sandwich:** Target (middle) & then converge

Big bang = bad (What was historically done)

How you integrate will largely determine your strategy.

Which to Choose?

Scheduling concerns.
Which parts of most critical/produce most concerns?
Component availability.
Infrastructure/hardware available?

Integration Testing: For Your Project

- Define a **testing strategy**
  - For your team
  - Across all teams
- Clearly define the **expectations** and agree to them
- What roles should do it?
- Assign point people
  - “Joe/Susie is responsible”

Verify the Need to Integrate First

- Do you need to keep the legacy system active? Do you even need to connect to it?
  - Don’t throw good money after bad.
  - Long term viability?
  - Security and reliability.

Discover the Integration Type

- Real-time integration: Alerts & updates information through integrate systems at the instant the change is made
  - Expensive
  - Risks?

Make Sure the Data is Clean

Eliminate redundant records
Fit into your system/process

Watch For System Overlaps

- Do new/existing apps already cover what the legacy system is doing?
  - The more you have, the more you have to maintain
  - Possible security vulnerabilities

Integration Testing Types

Dummy objects are passed around but never actually used. Usually they are just used to fill parameter lists.

Fake objects actually have working implementations, but usually take some shortcut which makes them not suitable for production (an in memory database is a good example).

Stubs provide canned answers to calls made during the test, usually not responding at all to anything outside what's programmed in for the test. Stubs may also record information about calls, such as an email gateway stub that remembers the messages it 'sent', or maybe only how many messages it 'sent'.

Mocks Objects pre-programmed with expectations which form a specification of the calls they are expected to receive.

Integration Testing Lifecycles

Test lifecycle with stubs:

1. Setup - Prepare object that is being tested and its stubs collaborators.
2. Exercise - Test the functionality.
3. Verify state - Use asserts to check object's state.
4. Teardown - Clean up resources.

Test lifecycle with mocks:

1. Setup data - Prepare object that is being tested.
2. **Setup expectations** - Prepare expectations in mock that is being used by primary object.
3. Exercise - Test the functionality.
4. **Verify expectations** - Verify that correct methods has been invoked in mock.
5. Verify state - Use asserts to check object's state.
6. Teardown - Clean up resources.

Recap

Integration concerns:
Types of integration testing:
“Faked Objects”

Recap

- Integration concerns: Technology, security, scalability, reliability, maintainability
- Types of integration
  - bottom, top, sandwich, big-gang (bad)
- “Faked Objects”
  - Dummy: Just fill parameters
  - Fake: Work, but take shortcut
  - Stub: Canned responses
  - Mock: More robust, pre programmed.

Release 1

http://www.se.rit.edu/~swen-343/project/projectdeliverables.html

Roles teams huddle

Responsibilities → document on Trello & Post on Slack **#general** as agreements by which you will be held accountable.

Integration Questions/Suggestions/Answers from *slides* → each huddle captures into separate Google doc to be shared back with silos and consolidated by **Team Coordinators**

Unknowns/TBDs/Risk → document as **Spikes** into Trello and assign individual(s) responsible

“Risks”