System Integration

Mini Case Studies

Course Summary and Wrap Up

Shawn A. Butler, Ph.D.
Senior Lecturer, Executive Education Program
Institute for Software Research
Carnegie Mellon University

Objectives

- Review some of the major points of the course
- Summarize lessons learned

Why are you integrating?

- Adaptable systems and processes
- Streamlined business processes
- Management information
- Support for electronic commerce
- Integrated security
- Replaceable components
- Reliable and recoverable systems
- Economies of scale

Integration Heuristics

- Simplify, Simplify, Simplify
- Don’t assume that the original statement of the problem is necessarily the best, or even the right one
- Build and maintain options as long as possible in the design and implementation of complex systems. You will need them
- In partitioning, choose the elements so that they are as independent as possible; that is, elements with low external complexity and high internal complexity

SDLC Phases

- Conception – What are the goals and objectives of the integrated system? What is the vision?
- Requirements Analysis – What are the requirements of the integrated system?
- Specification – What are the system detailed specifications?
- Design – What are the components and how will they communicate?

SDLC Phases cont'd

- Implementation – Integrate!
- Testing – testing and more testing
- Training and Support – Make sure the users know how to properly use what they paid for!
- Maintenance – Fix bugs and make changes as necessary
- Retirement – This almost never happens!

3 Models of Integration

- Presentation Integration – information is integrated through the user interface component
- Data Integration – information is integrated through a middleware layer that has business intelligence
- Functional Integration – information is integrated among databases or data sources

Common Services

- Naming Service
  - Every entity sees the entire system as the same set of objects
  - Define interfaces to entities
  - Users know how to find other system entities
- Security Service
  - All entities (principles) are authenticated with some degree of reliability
  - Access to resources are managed based on defined policies
  - Activities can be logged to ensure accountability and recovery
- Reliability – The ability of the system to provide necessary functionality despite unexpected events

Consistency

- Autonomous data sources are in a consistent state when a user can access data at any source and it will be the same
- After an update, how long before the system will be in a consistent state?
- What is the requirement? Immediately, within a few minutes, hours, or next day?
- Consistency requirements must be determined in distributed systems based on user requirements!
- The faster the consistency must be achieved, the more expensive the system

Why Do We Care about Consistency Issues?

- Modern Transaction Processing products ensure information consistency
  - Not always – Not all system objects covered
  - You may want to relax consistency requirements to improve performance
  - Cross domain transactions create integration problems
    - Different transaction processing products
    - Different assumptions about consistency

Two Applications with Overlapping Functionality

- Applications may have very similar, but slightly different functionality
- Application design has tightly coupled sub-components – Not easy to break apart
- Each sub-component may make assumptions about other subcomponents
- Redundancy in functionality adds maintenance overhead
- Sub-component not designed to be reused
- Especially problematic with redundant services

Possible Solutions for Overlapping Functionality

- Choose one application and extend functionality to meet total functionality
- Refactor one (or both) of the applications so that specific functionality can be extracted and then integrated
- Create API's to increase sub-component independence
- Wrap components and create new interfaces or clients
- Ignore functionality

Integration Guidelines

- Application Coupling – Minimize dependencies
  - Tightly coupled applications have known and unknown assumptions
  - Applications can evolve independently without problems
- Intrusiveness – Minimize changes in each application
  - Changes are often necessary
  - Tradeoff between intrusiveness and best integration design

Integration Styles

- File Transfer
- Shared Databases
- Remote Procedure Invocation
- Messaging Systems

Data Integration Mistakes

- Creating yet another database
- Waiting for the data analyst to finish developing the perfect schema
- Implementing the perfectly normalized schema
- Assuming the data exists as described in the documentation
- Testing without an sufficient set of real data
- Assuming that one site is a good representation of data at all sites

Relational Databases

- Database normalization rules are designed to prevent anomalies and inconsistencies in databases
- Database normalization rules, strictly applied, may introduce inefficiencies in database design
- All databases have various schemas that work well for the their application, but don't combine well into an efficient schema for all applications
- Database design is always a tradeoff among application performance optimization

Security Integration

- Authentication, access control, and auditing are the fundamentals of system security
- Integration of security services is difficult and takes considerable planning
- Integration of security services may introduce more risk than the risk of each component
- Fundamental element of security is risk

Middleware

- Messaging is one of the most popular middleware technologies for integrating disparate applications
  - Asynchronous communication, but can be designed to do synchronous communication
  - Different types of MOM
    - Point to point message queuing
    - Broadcasting
- CORBA not as popular, but still out there
- Lots of middleware options
- Transaction Processing Monitors ensures ACID properties in heterogeneous environments

Accessing the System

- On the network
  - Local Area Networks
  - Standing Virtual Private Networks
  - Wide Area Networks
- Remotely
  - Remote Desktop Protocol (RDP)
  - Terminal Services
  - Virtual Private Networks
- DNS, routers, and Active Directory are key to sharing information across a network

Clients – Fat or Thin?

- Provide adequate interactive performance
- Ensure the integrity of system updates
- Minimize risk of sensitive information exposure
- Minimize network activity
- Enable disconnected user activity

Enterprise Client Devices

- Specialized devices are becoming an essential part of the enterprise solution
  - Cell phones/PDA's
  - Mobile media devices
- Protection of these devices is a significant challenge
- User interfaces change among different devices
- Wireless connections are expanding

Summary

- There are many aspects to integrating a system
- System engineers have to understand a broad spectrum of hardware and software technologies
- Good system engineers are good because they have made many mistakes, or have seen many mistakes
- Your ability to anticipate changes, see how the system fits together, and pay attention to detail are good skills for a system engineer
- When on a project, use your journals to remember the things that can go wrong!