Planning and Surviving a System Upgrade – Lessons Learned

Session 335

AACRAO 2007

Sam Conte, University of Pittsburgh
Paul Robson, University of Michigan
Sue Van Voorhis, University of Minnesota
Necessary Parts

Establish possible upgrade dates
  ● Include all parties affected

○ Agreed Upon Methodology

○ Detailed Project Plan
  ● Establish budget and resources
  ● Develop Freeze Strategy
    ○ Analyze required maintenance
    ○ Project exceptions
  ● Applied previous upgrade lessons learned

○ Audit Review
Upgrade Project Discussion

- Is the upgrade necessary?
  - Later version better
  - Impact
  - Patches (continuously)
  - User Fatigue

- Upgrade date selected
  - Pros and Cons (impacts)
    - Students vs. Employees vs. Finance
      - Orientation
    - Fiscal cycle vs. academic cycle
    - New functionality

- Cost
Project Initiation
(Appetizer)

- Assign experienced project manager

- Establish budget and controls
- Define “issue resolution” process
- Develop communication plan
- Define change management process
- Define risk management process
- Conduct initial risk assessment

- Kick off meeting – stakeholder review
Project Charter Development

- Develop project charter
- Incorporate stakeholder feedback
- Apply lessons learned
- Charter review and signoff

Functional Input Necessary!
Current System Change Management

Freeze strategy

- Develop process for freeze exceptions
- Document exceptions
  - Mandates – senate, federal and state
  - Current projects – parallel development etc.
  - Mandatory processes – tax updates, financial aid regs
- Analyze required maintenance
  - Identify – tax releases, FA regs, critical fixes, bugs
  - Define implementation points
  - Incorporate exceptions into upgrade
Analysis of Project Strategy

- Determine prototyping strategy
- Define archiving strategy
- Define security strategy
- Define query strategy (who do retrofits)
- Define testing strategy
  - system, parallel, performance
- Coordinate quality assurance

Review any release notes and documentation
Customization Review

- Identify retrofits
- Complete business requirements
- Personal Approach
  - Meet with users - findings and functionality
  - Determine requirements/mods needed
- More analysis items
  - Pair functional and technical staff
  - Gap/fit decisions
  - Document new requirements and mods
Design
(Significant Time Needed)

- Review Standards
- Menu structure evaluation
- Development planning
- Architectural design
- Functional design (new development)
- Technical design (new development)

Final Time, Cost and Schedule Estimate
Development
(Main Course)

- **Implementation planning**
  - Go/No Go planning
  - View-only Instance – upgrade conversion
  - Data Warehouse access
  - Post implementation support

- **Training strategy**
  - End user – plan and schedule
  - Delta Training - Pitt
  - eLearning, web based
  - Training instance
  - Transformation workshops - UM
Application Integration

- Define technical requirements
  - Bolt on applications integration
    - Each area identifies impacts
  - Interfaces
  - Testing/performance
  - Data warehouse – table structure
Code/Unit Test Retro Fits

- Records objects
- Reports
- Database names
- Triggers
- Database links
- Workflow
- Scripts
- Code changes

Tchie Info
New Development

- Code new modification
- Code review
- Unit testing (define)
- Migration review
- Create test plan and test cases

Update migration spreadsheet

Techie Info
Testing

- Create and/or update test plan and cases
  - Test retrofits
  - Test new functionality
  - Test vanilla software
  - Test new development
  - Test integration of outside applications
  - Develop load test plan/analysis

Business Analyst/functional
Parallel Testing

- Payroll
- Student bill
- Admissions reports
- Various reports
Performance Testing

- Execute
- Analyze
- Evaluate
- Tune

Repeat if necessary.

VERY important!!!!!
Enterprise Testing

- Security testing
- Cross functional testing
- Acceptance testing
- Application upgrade testing
- Database upgrade testing
- Internal and external interfaces
Technical Infrastructure

- Server testing
  - Generic access
    - Hardware setup
    - Software setup
  - Central authentication
- Upgrade strategy with client software
  - Other applications
- Initial software install

_Techie Info_
Upgrade Pass

- Initial upgrade pass
  - Part 1 – Create instance for developers
  - Part 2 – Data Conversion
  - Part 3 – Merge above to create instance
- Practice 1 - w/90% of mods complete
- Practice 2 – account for deltas
- Practice 3 – account for deltas
  - Dress rehearsal

Techie Info
Upgrade Begins
(Dessert)

- Signoff to begin upgrade by business
- Communication to all affected
- Begin upgrade
- Shutdown activities
- Start upgrade activities
- Functional verification
- Go live
- Post upgrade issues
- Transition to maintenance
- Celebration
- Post implementation review – lessons learned
Lessons Learned

- **Schedule upgrades for February (UM)**
  - Fall – too busy
  - Summer – too close to start of term
  - Spring - too busy for student administration

- **Determine accurate project timeline**
  - Last upgrade too long
  - Current upgrade too short
    - Increased use of functionality
    - Increase in mods results in more retro fits & testing
Planning and Management

- Contact other schools about issues
- Investigate impact of upgrading
  - Application, tools, database simultaneously
    - If issues occur, difficult to determine the cause
- Develop formal prototyping phase
  - Complete before project begins
- Create communications team early
  - Create communications plan early (internal/external)
  - Create crisis communications plan
Internal Work Plan

- Schedule testing activities for no more than 32 hours per week
  - Other issues need attention

- Schedule security testing

- Investigate listserv with archive option instead of a wiki
Testing

- Allow sufficient time for unit testing
  - Functional testing began finding out the unit testing was not completed by developers
  - Define unit testing vs. functional testing
- Create list of items for functional verification so all items are available
- Communicate functional verification status to stakeholders
- Use a tool to run test plans – Quality Center
  - Allow shifting of testing to other staff
Testing cont.

- Create and document sources of test data
- Allow sufficient time for system testing
- Develop detailed plan for cross-functional testing
- Run load test that will reflect true numbers
- Create performance testing instance
  - Develop strategy to create accurate results
- Review queries for accuracy
- Conduct more process scheduler testing
- Create detailed acceptance testing plan
Training

- Involve training team in project timeline discussions early
- Create training instance early
- Allow time to update documentation
- Create consequence for neglectful user
Technical Infrastructure/Upgrade/Project Close

- Single ownership and coordinate between upgrade software and other software/applications
- Security matrix is needed
- Improved communication between all impacted IT staff not just those upgrading
- Responsive team - address issues & bugs
- IT staff take ownership of problems
  - Database, developers, hardware support etc.
- Develop contingency plan

Techie Info
Additional Items

- Account for patches and fixes sent by vendor
- Develop measures for testing completion
- Analyze table structure differences
Goals

- Eliminate modifications
- Meet timelines/deadlines
- Minimize consultant usage
- Use single tracking tool
- Technical and function verification – no issues
- Detailed reusable test plans
- Synergy
- Teamwork
- Success
Questions

Contact Information:

Sam Conte          sdconte@pitt.edu
Paul Robson        probins@umich.edu
Sue Van Voorhis    vanvo002@umn.edu