Ready to Upgrade?
Migration Best Practices

Mike Atherton, xTuple DC

Saturday, October 12, 2013

Migration: Opportunity

- View a migration as a small implementation
- View a migration as an opportunity to address training deficiencies
- View a migration as an opportunity to implement new business practices
- View a migration as an opportunity to implement new functionality
- View a migration as an opportunity to do a little house cleaning

Technical Risk

- Database Reorganization
  - Deprecated columns
  - Deprecated functions
  - Updated functions
  - Deprecated tables
  - Changes to API.views

Keep in mind: An enhancement (report, MetaSQL, function change, scripted app) made on release X will expect to see the same database structure as X after you have migrated to release Y. Usually this works. But sometimes....

Technical Risk

- Toolbox changes
  - Changes to xTuple Qt scripting toolbox
  - Screen layout changes that impact scripts
  - Updater changes
  - CSVimp changes
  - OpenRPT changes
  - Supported version of Postgres

Technical Risk

- Package Compatibility
  - Refer to this link to ensure compatibilty:

    http://www.xtuple.org/compatibility-matrix
  - xTuple modules
  - xTuple Connect
  - xTuple Updater
  - Third Party Applications
  - Postgres

Human Factors

- Additional / Enhanced Functionality
  - Confusion
  - Lost opportunity to address existing requirement
  - As simple as: new column on existing screen
  - As big as: entirely new capability

Other Risks

- Don't forget:
  - Releases implement new privs that if not addressed can lockout users from existing functionality
  - A bad migration reduces confidence in the system
  - But, a good migration is seamless
    - •Generates excitement over new features and functions
    - •...but only if users know about them

Plan, Pilot, Migrate

- These are the major steps
  - Plan
    - •Develop a written plan
  - Pilot
    - •Test the migration process
    - •Use business process flows to
      - –Discover technical problems
      - –Discover functional issues
      - –Prepare and train users
  - Migrate
    - •A good plan makes the migration process
      - –Low risk
      - –Fast
      - –Seamless

Plan

- Develop a written plan that includes these sections
  - Pre-migration to-dos (examples):
    - •Switch to admin mode
    - •Backup production DB
    - •Disable trigger xyz
  - Migration scripts in sequence with expected duration (ex: std41xto42x.gz)
  - Post migration to-dos (examples):
    - •Application of remediated components such as:
      - –Reports
      - –DB objects
      - –Scripts
      - –Screens
      - –MetaSQL
    - •Switch out of admin mode

Plan

- Develop a written plan that includes these sections (cont'd)
  - Order-to-Cash Flow process flows:
    - •Sales to cash receipt
    - •Purchase to payment processing
    - •Work Order to close
    - •Touches all custom functionality
  - Customized components testing:
    - •Reports above grade 0
    - •MetaSQL above grade 0
    - •Scripts (use order by > 0 to identify custom/modified scripts)
    - •Screens (use order by > 0 to identify custom/modified scripts)
    - •DB objects (functions, triggers)

Plan

- Develop a written plan that includes these sections (cont'd)
  - Packages List
    - •Ensure all have been tested in a migrated pilot
  - EDI Profiles
    - •Test in a migrated pilot
    - •Cancel all scheduled jobs in the migrated pilot
      - –Avoid sending “live” looking transmissions to recipients
    - •Test all EDI profiles in migrated pilot
      - –Avoid sending “live” looking transmissions to receipients

Piloting

- Use the written plan to record issues discovered in the migration process
  - Update the written plan with how each issue is addressed
  - Fixed / Staged
- Use the written plan to record issues discovered in the business process flow piloting process
  - Discovery process
  - Validation process
  - Training opportunity
  - Record how issues are address in the written migration plan

Piloting

- Dry Run Migration Pilots
  - Use the written plan to execute “dry run” migration
- CAUTION!!!
  - Do not count on users with access to migrated pilots to actually validate their functional area
  - Pull them into a pilot session
  - Execute it with them
  - Validation process
  - Training opportunity
  - Record how issues are address in the written migration plan

Staging

- Stage Remediated Components
  - Fix components in pilot and stage externally in a migration folder
  - Keep the final pilot where fixes reside as a reference DB
  - Load fixed components in live DB with:
    - Grade < highest grade
    - Order < highest order AND inactive
  - Update Migration Plan To-do's with staged items that require promotion
- Fix and Backport
  - Fix in migrated pilot
  - Test in live DB and if it works, deploy

Migration

- Planning makes the process go smoothly
  - With good preparation and a good plan it has been done at lunch
  - Better choice is after close of business
  - Plan for how the new client will be deployed
- Backups
  - Create two before starting:
    - Backup to file
    - Backup on the production server to a “pre-migration” DB
      - This backup is on the server ready to use in a pinch
      - But with a good plan you will not need it

Timing

- Frequency
  - Annually if not more often
  - Less frequent migrations simply compound the planning process
    - Increases risk
    - Takes longer
    - Bigger jump for users

Contact

Mike Atherton

xTuple Evangelist & Director of Training

Mike@xTuple.com