

Practical Approaches to Large Model Management

Tricks and Tools of the Trade

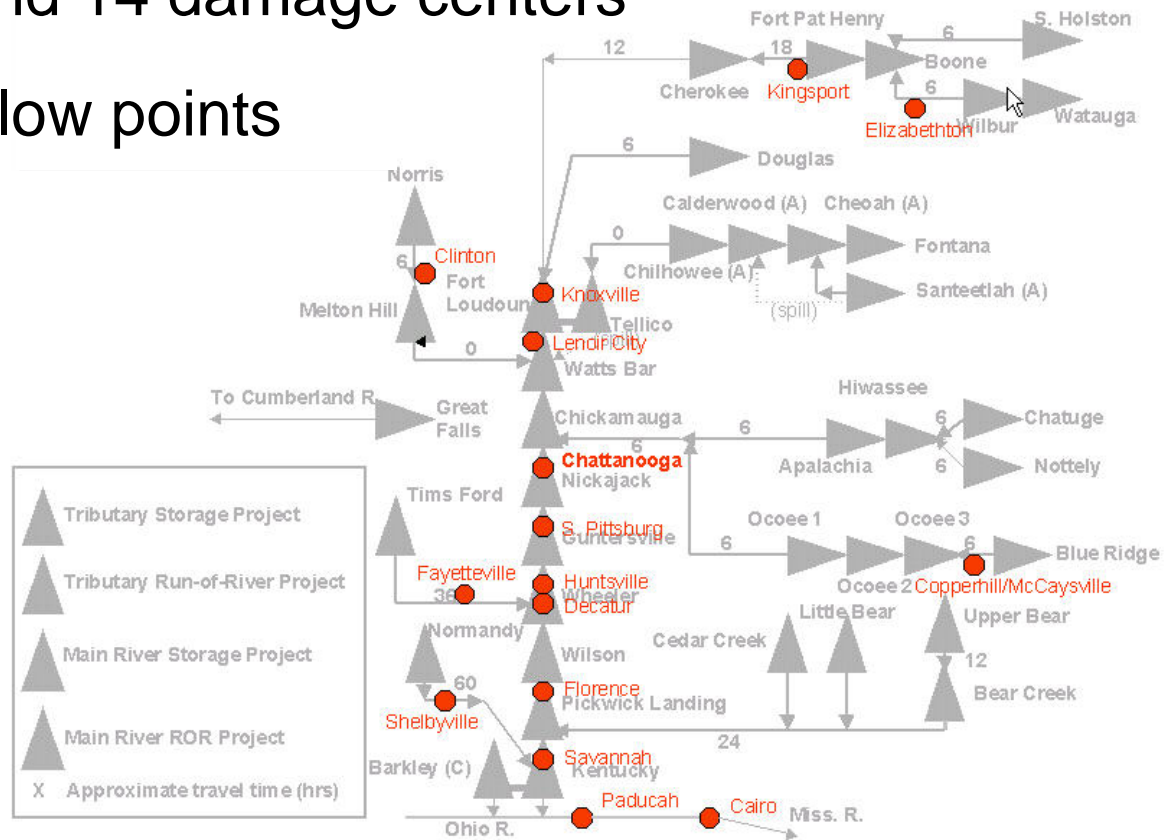
James VanShaar

Riverside Technology, inc.

A LARGE, COMPLEX MODEL

🔴 37 dams and 14 damage centers

🔴 55 local inflow points



A LARGE, COMPLEX MODEL

- 37 dams and 14 damage centers
- 55 local inflow points
- 99 years at 6 hour timestep: ~144k timesteps

A LARGE, COMPLEX MODEL

- 37 dams and 14 damage centers
- 55 local inflow points
- 99 years at 6 hour timestep: ~144k timesteps
- 69 historic storms scaled 1.5x, 2.0x and 2.5x

A LARGE, COMPLEX MODEL

Major Concerns

- Run-time
- Model size
- Accuracy of policy representation
- Decision tracking: debugging, calibration, reproduction
- Extensibility to alternatives

Trick 1: Careful Rule Design

Generic Algorithms

Generic Tributary Algorithms

- Applied to virtually all non-sloped power reservoirs
- Foundation of all operation policy
- Quarantined deviating code

Mainstem Fixed Rule (sloped-power reservoir)

- Acceptable discharge vs. pool elevation operational points

The image displays three overlapping screenshots of the software interface. The top window, titled "Ruleset Editor - '04-22-03TA_0a.rls'", shows a table of policy groups:

Priority	On	Name	Type
	✓	Tributary_FinalOutflowAndQuitTest	Policy Group
	✗	GenericTrib_Secondary-UpperMainstem	Policy Group
	✓	GenericTrib_Secondary	Policy Group
	✓	Mainstem_QuitTest	Policy Group
	✗	LowerMainstem_FixedRules	Policy Group

The bottom-left window, titled "Policy Group Editor - '04-22-03TA_0a.rls : GenericTrib_Pre'", shows a table of rules:

Priority	On	Name
32	✓	CalcPreliminaryOutflow
33	✓	NonSOGMode_Debug
34	✓	SetDecisionTrigger
35	✓	Qc'Zero_FloodModeReservoirs
36	✓	Qc'_FloodModeReservoirs
37	✓	Qc_RecoveryModeReservoirs
38	✓	QsogToOutflow
39	✓	Qsog_Qf
40	✓	FloodTriggerRule
41	✓	Set_RemainingBlendSteps

The bottom-right window, titled "Rule Editor - '04-22-03TA_0a.rls : GenericTrib_Preliminary : SetDecisionTrigger'", shows the rule body:

```
FOR EACH (OBJECT Reservoir IN ListSubbasin ("Tributary Reservoir")) DO
  DataObjectName (Reservoir) = "DecisionTrigger" []
  = IF (NOT InFloodMode (Reservoir)) THEN
    IF (PoolElevationAboveSOGPlusTolerance (Reservoir)) THEN
      2.00000000
    ELSE
      1.00000000
    ENDIF
  ELSE
    IF { TopOGatesExceedanceExpected (Reservoir, CriticalTOGOOutflow (Reservoir)) } THEN
      IF (PreviousPoolElevation (Reservoir) <= TopOGates (Reservoir)) THEN
        8.00000000
      ELSE
        8.50000000
      ENDIF
    ELSE
      IF (TopOGatesReachedTriTimestep (Reservoir)) THEN
        9.00000000
      ELSE
        8.00000000
      ENDIF
    ENDIF
  ENDIF
ENDIF
```

The "Execute Rule Only Where:" field contains: DecisionTriggerNotSet (TestReservoir 0)

Trick 1: Careful Rule Design

Generic Algorithms

Results of rule set design

- 🔴 Carefully tested, compact, reused code base
- 🔴 Eliminated re-firing of rules
- 🔴 Decision variables stored
- 🔴 Limited re-resolution of objects
- 🔴 Individual policy relegated to parameters, not logic

Trick 2: System Segmentation

Space

The image displays a screenshot of the RiverWare 4.2 software interface, illustrating system segmentation. The main window shows a network diagram with nodes and connections, categorized into three segments:

- Upper Tributary:** This segment includes nodes such as SystemDummyReach, SystemDummyReach, SouthHolstonReachData, SouthHolstonData, SystemData, ObservedData, FortPatrickHenryELFData, BooneData, SouthHolstonReach, SouthHolston, ElizabethtonToBooneData, ElizabethtonELFData, WakaugaData, HolstonToCherokeeData, KingsportData, FortPatrickHenryELF, Boone, PseudoBoone, PseudoBooneData, ElizabethtonToBoone, Ocoee1ReachData, Ocoee1Data, Ocoee2ELFData, Ocoee3SpillReach, Ocoee, Ocoee1Reach, Ocoee1, Ocoee2SpillReach, Ocoee2, Ocoee2ELF, Ocoee2Confluence, Ocoee3, Ocoee3, TimsFord, TimsFordReach, FayettevilleELF, TimsFordData, TimsFordReachData, FayettevilleELFData, SanteetlahData, and Santeetlah.
- Upper Mainstem:** This segment includes nodes such as Norris, Cherokee, Douglas, Fontana, SanteetlahTubIn, FontanaSanteetlahTurbine, CheoahELF, ApalacheeELFData, ChickamaugaData, HolstonToKnoxvilleData, OakdaleELFData, ApalacheeReachData, ChickamaugaData, KnoxvilleELFData, Pseudo_FloodAndInflData, CalderwoodELFData, ClintonELFData, MaryvilleELFData, SeviervilleELFData, CharlestonELFData, ClintonToMeltonHillData, MeltonHillELFData, SystemData, ChatugaData, NickajackData, SystemDummyReachData, ChatugaReachData, DouglasData, NickajackData, SystemDummyReachData, NorrisData, MeltonHillData, TellicoData, CheoahELFData, FortLoudounData, WatsBarData, CherokeeData, FrenchRoadToKnoxvilleData, NickajackReachData, ObservedData, MeltonHill, MeltonHill, FortLoudoun, TellicoCanal, Chatahoochee, Calderwood, CheoahSanteetlahSpill, SanteetlahSpill, MeltonHillOakdaleConfluence, WattsBar, ApalacheeReach, SouthChick, ApalacheeELF, Ocoee1Reach, ApalacheeELF, Hwassoee, ChatugaReach, Chatuga, NottelyReach, Nottely, Nickajack, SouthChickamaugaCreeELF, and SystemDummyReach.
- Lower Mainstem:** This segment is represented by an aerial photograph of a large dam and reservoir.

Two inset images provide visual context: the top right shows a dam with a reservoir, labeled "Non-Power Tributary", and the bottom left shows a dam with a reservoir, labeled "Lower Mainstem".

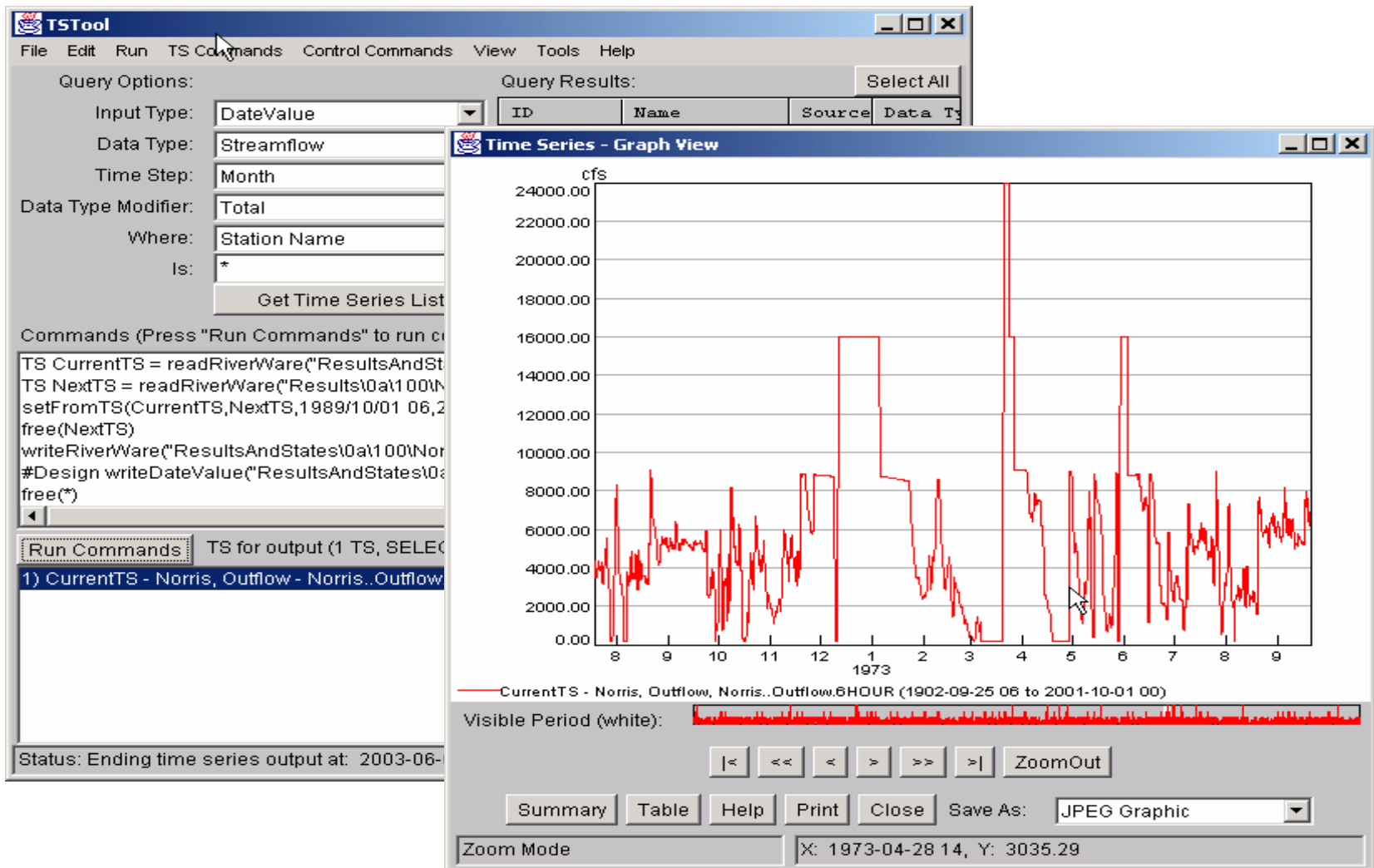
Trick 2: System Segmentation

Time

- Initial conditions
- Archival and access to state variables
- Control of sub-periods

Tool 1: TSTool

Data Management



Tool 2: Perl Scripting

Control and Data Management

```
# The following array contains paired values X,Y where X will be used to
# time series handling scripts and model periods.
@dateSwaps = ( "1902_1903", "1903_1913", "1913_1923", "1923_1933", "1933_
1953", "1953_1963", "1963_1973", "1973_1983", "1983_1989", "1989_
1993" );
#@dateSwaps = ("1947_2001");

$DesignRun = "100";   #<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$AltNum = "0a";
$RunNumber = 0;

$Initialize = "k:\PROJECTS\A636_TUA\PHASE_3\DMITSTool_TribsAllInitialize
$NewInitialize = join(" ", $Initialize, $AltNum, $DesignRun);

# The following is not needed for Tributaries
##$Dval = "k:\PROJECTS\A636_TUA\PHASE_3\DMITSTool_MainstemTB_DVal";
##$NewDval = join(" ", $Dval, $AltNum, $DesignRun);

$KnownInflows = "k:\PROJECTS\A636_TUA\PHASE_3\DMITSTool_KnownInflows_forB
$NewKnownInflows = join(" ", $KnownInflows, $AltNum, $DesignRun, "_T");

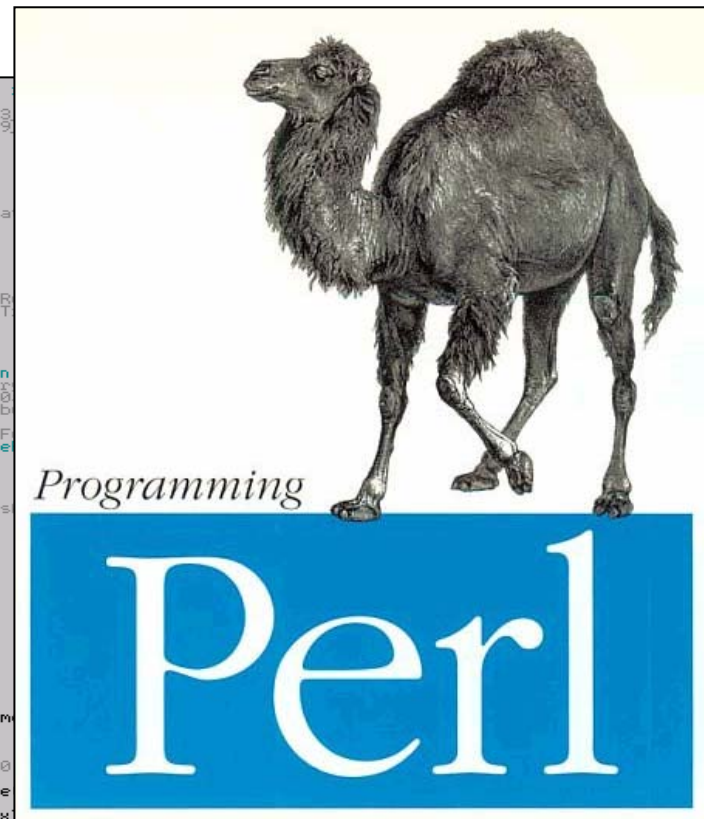
$RCL = "k:\PROJECTS\A636_TUA\PHASE_3\DMI\BATCH\TribsAll.rc1";
##$RCL = "k:\PROJECTS\A636_TUA\PHASE_3\DMI\BATCH\TribsAll.rc1POST";
$NewRCL = join(" ", $RCL, $AltNum, $DesignRun);
# Note the backslashes are required to produce two backslashes in
$BaseModel = "K:\PROJECTS\A636_TUA\Phase_3\Models\Tributar
Rules = "k:\PROJECTS\A636_TUA\Phase_3\Models\04-22-03TA_0
$BaseSaverModel = "C:\PROJECTS\A636_TUA\PHASE_3\MODELS\Tribs

$updateResults = "k:\PROJECTS\A636_TUA\PHASE_3\DMITSTool_TribsAllUpdateF
$updateResults = "k:\PROJECTS\A636_TUA\PHASE_3\DMITSTool_TribsAllUpdate
$NewUpdateResults = join(" ", $updateResults, $AltNum, $DesignRun);

$TSTool = "W:\PROGRA~1\B1I\RIVERT~1\BIN\TSTOOL.BAT";
$RiverWare = "C:\PROGRA~1\CADSWES\RIVERW~1.2\RIVERW~1.EXE";
$RiverWareOutput = join(" ", "TribsAll", $AltNum, $DesignRun, "_kTribs

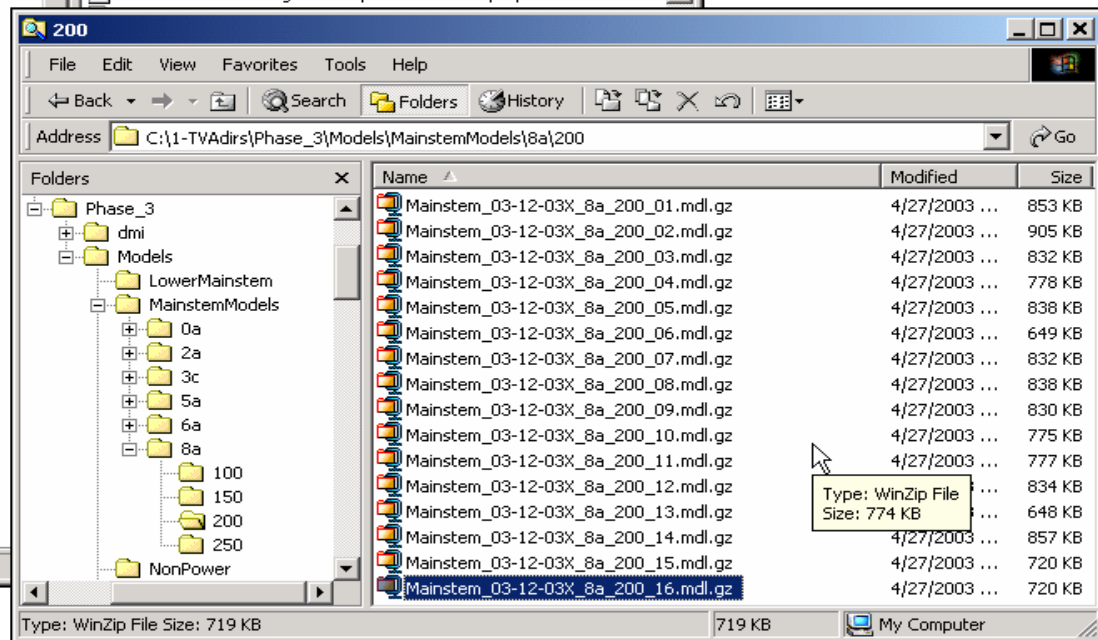
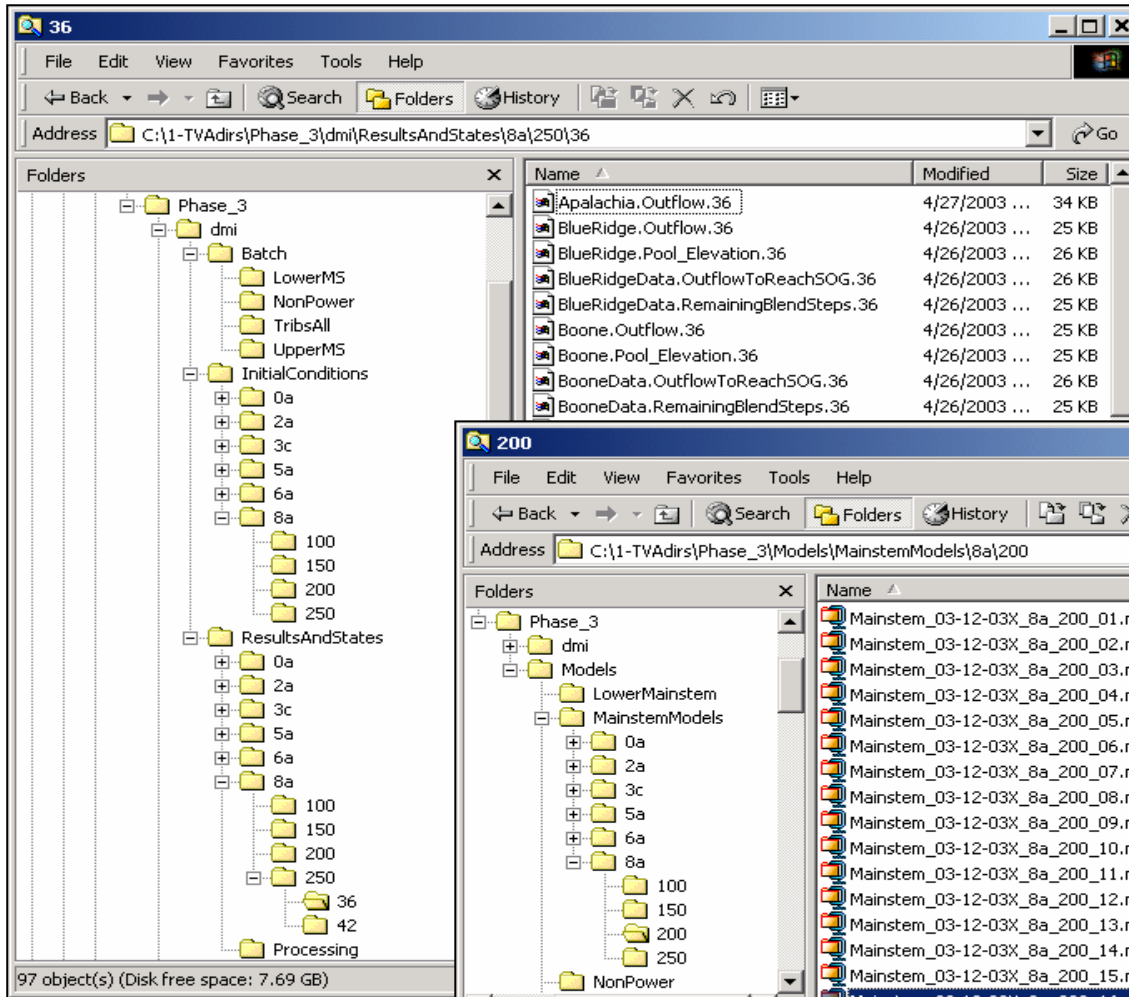
$ftpComm = "k:\PROJECTS\A636_TUA\PHASE_3\DMI\BATCH\ftpgcomm";
$NewFTP = join(" ", "ftpComm", $AltNum, $DesignRun, "_TribsAll");

$stillGO = 1;
if ( 1 ) {
  foreach $year (@dateSwaps) {
    if ( $stillGO ) {
      print "year\n", $year ;
      $RunNumber++;
      # Set up the dates
      @xxTime[0] = "@Fld[01]/10/01 06";
      $xxTimePrexx = "1901/12/31 24";
      for ( $i = 1; $i <= 17; $i++ ) {
        print @xxTime[$i] = SubtractHoursFromDate( @xxTime
        print "\n";
      }
      if ( @Fld[1] < 2001 ) {
        print $xxTimeEndAnalysisWindow = "@Fld[1]/09/30
        print "\n";
        print $xxTimeEndModelRunPeriod = AddHoursToDate
        print "\n";
        print $xxTimeEndModelLELFs = AddHoursToDate( $xx
        print "\n";
      }
      else {
        print $xxTimeEndModelRunPeriod = "@Fld[1]/09/22 18";
        print "\n";
        # Note the next line gets ready for a SubtractHoursFromDate of the sar
        print $xxTimeEndAnalysisWindow = AddHoursToDate( $xxTimeEndModelRunP
        print "\n";
        print $xxTimeEndModelLELFs = AddHoursToDate( $xxTimeEndModelRunPeriod
        print "\n";
      }
      # Update TSTool Initialization command file
    }
  }
}
```



Tool 3: Directory Structure

Data Management



Trick 3: Integration of Tools

Control and Data Management

Control Algorithm: For each successive run period--

- Modify TSTool and RiverWare batch control files
- Run TSTool initialization commands
 - Access archived data
 - Locate RiverWare input in expected directory
- Run RiverWare using its control file
 - Import data
 - Simulation
 - Export data
 - Save model with new name
- Run TSTool archival commands
 - Store results in archive time series files

Results of Approach

- Flexibility
 - Debugging
 - Event isolation
- Run-time
- Consistency throughout alternatives
- Built-in archival of runs / models / decisions
- Elimination of model size concerns

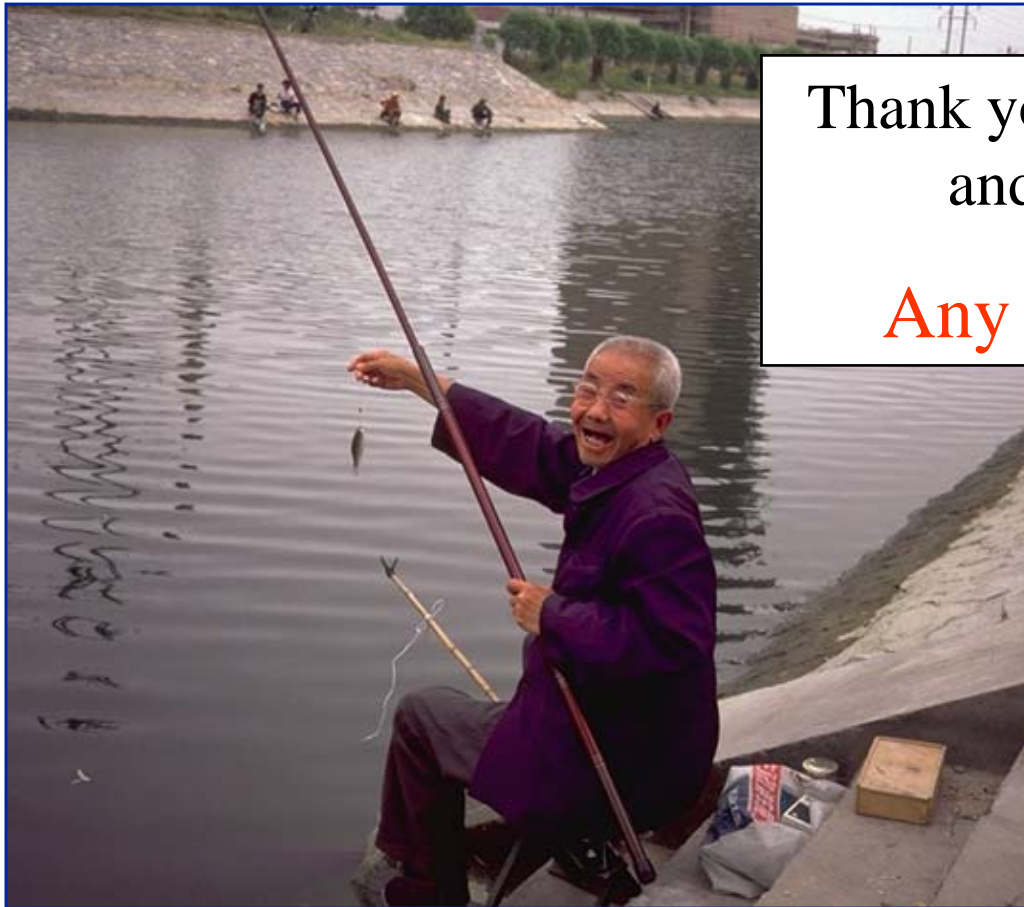
Looking to the Future

- More large models
- Water Accounting
- Optimization

- Real-time operation
- Probabilistic forecasting

- Post-processing Product Generation

Conclusion



Thank you for your time
and attention.

Any Questions?

IDIQ Contractor for A&E Services (2003)

NOAA Partner for Advanced Hydrologic
Prediction Services (AHPS 2003)

GSA Schedule (2004)