Replacing a Legacy

• MONTE
  – Mission Analysis, Operations and Navigation Toolkit Environment
  – Developed to modernize, upgrade, unify JPL’s navigation, maneuver, and mission design software (DPTRAJ/ODP/MASL)
    • Software developed beginning in the ‘60s with over 30 years of proven track record
  – Goals
    • Exploit advances in computational technology
    • Retire risk associated with old technology
    • Free ourselves from the constraints of the old technology
      – Use OO, modern development processes, modern development tools.
    – MONTE has achieved these goals and today is JPL’s premier navigation and mission design software.
Development Considerations

• Modern open standard OO language
  – C++ provides compiled OO with benefits of C

• Exploit Open Source

• A scriptable toolbox OO interface
  – Python to present user connection to C++
    • Extensible, worldwide open source community, platform independent

• Strong balance development process
  – CMMI maturity level 3
  – Development team was JPL’s pathfinder in CMMI
MONTE Architecture

- Python Environment
  - Applications
    - Mathematics Processing
      - Measurement
        - Physics
        - Data Store
        - Data Service
  - Optimization and Navigation Workflow
  - User Controlled Variables
    - Range, Doppler, VLBI, Optical
    - Time, Ephemeris, Orientation, Forces, Coordinate Systems
  - Persistent Objects
    - Tracking Data, Earth Orientation, Leap Seconds, SPICE kernels, etc
  - Syntax, Third Party Capabilities, User Specified Objects

Applications

- Users need high level capabilities for graphical manipulation and to provide common scriptable workflows.
  - UI system
  - Multi-leg Trajectory Optimization
  - Trajectory Differential Corrector
  - Access to the Horizons Small Body Ephemeris System
Applications

- Residual viewing and editing
- Landing site statistical hazard avoidance
- Launch contour analysis
- 3-D rendering and analysis
MONTE Ecosystem

• Documentation
  – Documentation cross-linked, web-based system
MONTE Ecosystem

- Documentation
- Tutorials
- User Guides
- Tested Examples
- Videos

```python
( inside the DivaPropagator definition )

# First create the integration state, and set parameters to user-defined
# or default values. The actual state to be propagated will be added at
# a later time.
istate = M.IntegSetup( boa )
    istate.setStateTol( StateTol )
    istate.setMassTol( MassTol )
    istate.setFrameTol( FrameTol )
    istate.setTimeTol( TimeTol )
    istate.setUserTol( UserTol )
    istate.setPartialTolScale( PartialTolScale )
    istate.setTimeFrame( IntegTimeFrame )
    istate.setResetStm( ResetStm )
    istate.setStateForces( Forces )

# Create the propagator with the empty state, and set tolerances.
obj = M.DivaPropagator( boa, Name, istate )
    obj.setMinStep( MinStep )
    obj.setMaxStep( MaxStep )
    obj.setRelativeParTol( RelativeParTol )
    obj.setCacheSize( CacheSize )
    obj.setDiffLinesPerLeg( DiffLinesPerLeg )
```

- Most text/equations are embedded in the source code where
  the capability is implemented
- Complete doc strings in Python interface.
MONTE Ecosystem

• Process
  – Unit Test Requirements
    • All functions require testing
    • Code coverage
    • All tests configuration managed
  – Style Requirements
  – Defect Tracking
    • Bugzilla
  – Software Metrics
  – Daily Clean Night Build and Test
  – Defined Release Process
  – Defined Scope Management Process
  – Stakeholder Communications
    • Bulletin Boards
    • Participation in bi-weekly Mission Designer and Navigator Meetings
Getting It Right

• Test System
  – Testing is extensive
    • 700 kslc deliverable
    • 1400 kslc of test code
  – User design/developer implemented system tests

• Where capabilities overlap round-off agreement with legacy software

• User testing of new features that are then incorporated into the system tests

• Defect response
  – Write a test that demonstrates the problem
  – Fix the code, see that that test passes
  – Run all regression tests
Operations and Adoption

- Adoption by mission required a push by management
  - Meetings every 2 weeks to analyze progress