### Motivation for undo
- Human error is a major impediment to dependability
  - largest single contributing factor to outages
- Undo is a recovery mechanism well-matched to coping with human (and non-human) error
  - tolerates inevitable errors
  - harnesses hindsight and provides retroactive repair
  - ~70% of human errors are immediately self-detected
- Supports trial & error exploration of complex systems
  - allows operators to learn from mistakes

### The 3R undo model
- **Undo** == time travel for system operators
  - **Rewind**: roll system state backwards in time
  - **Repair**: change system to prevent failure
    - e.g., edit history, fix latent error, retry unsuccessful operation, install preventative patch
  - **Replay**: roll system state forward, replaying end-user interactions lost during rewind
- **Challenges in 3R model**
  - external inconsistency (introduced by repair/replay)
  - drawing the undo boundary (what state is preserved)
  - providing undo at multiple granularities (user/sys/...)

### Challenges in 3R model
- Undo requires replay-acceptable histories!
  - A history is replay-safe if:
    - it contains only safe operations, OR
    - it contains unsafe or deleted operations
      - the history can be made replay-safe by inserting appropriate compensating actions
      - these histories have acceptable visible inconsistency
      - undo requires replay-acceptable histories!

### Modeling undo logging
- Application interface defined by a set of verbs
  - verbs define actions on logically-named state entities
  - e-mail examples:
    - deliver, fetch, set flags, delete, refile, create folder, ...
- **Operations are instances of verbs**
  - reflect actual user/operator interaction
- **The undo log is a history of operations**
  - during repair, the history may be modified
  - and other changes may be made to the system that aren’t reflected in the history
- Fringe benefit: shipping undo log enables easy georeplication

### Modeling operations
- Each logged operation is modeled by:
  - a verb specifying the action
    - a set of state entities needed to carry out the action
    - a set of preconditions over the state entities
      - all preconditions and no non-trivial ones
    - an indication of which state is modified
    - an indication of which state is externalized
    - a time specifying when results are externalized
    - used to classify operation as safe or unsafe
  - used to determine if unsafe state is externalized
    - externalized unsafe state => external inconsistency

### Making histories replay-acceptable
- **Step 1**: identify externalized unsafe operations
- **Step 2**: insert app-specific compensations
  - explanatory ("msg was deleted because it had a virus")
  - repairing (create "lost&found" to replace missing folder)
- A dependable undo system must be able to make any history replay-acceptable:
  - operation templates (verbs) must be specified correctly
    - all needed preconditions and no extraneous ones
  - compensations must exist for all precondition violations
    - precondition and compensation logic must be correct
      - model identifies cases for exhaustive testing

### Model benefits
- Simplifies reasoning about undo inconsistency
  - expressed in terms of preconditions & compensations
- Provides greater confidence in undo
  - by construction, if preconditions are correct and compensations exist, all scenarios will produce acceptable external consistency
  - declarative specifications of verbs, preconditions, and compensations are easier to write and check
  - model provides guidance for exhaustive testing
- Provides framework for general implementation
  - Implicitly defines properties of applications that can be wrapped for undo
    - verb interface, GUID-namable state, complete API, tolerance for mild/explained inconsistency

### Foundation: undo system structure
- An undoable system consists of:
  - an application with a well-defined, non-procedural user interface (a service)
  - a stable storage layer supporting time travel
    - snapshots, backups, non-overwriting/log-structured FS
  - an undo wrapper that logs and replays user/operator interactions with the application
  - A dependable undo system must be able to make any history replay-acceptable
    - declarative verb specs and general mechanism layer
    - declarative specifications of verbs, preconditions, and compensations are easier to write and check
    - extend model for automated consistency analysis
      - enable "what-if" analysis of repair impact
      - requires expanding verb definitions with specification of expected state changes
      - could compare with dynamic analysis to identify bugs?

### Status and future directions
- **Status**
  - continuing model development using e-mail as driver
    - next step: try to better formalize compensations
  - restarting implementation to follow the model
    - more declarative verb specs and general mechanism layer
- **Future directions**
  - extend model for automated consistency analysis
    - enable "what-if" analysis of repair impact
    - requires expanding verb definitions with specification of expected state changes
    - given buggy software and arbitrary repairs, automated analysis would be just a hint
    - could compare with dynamic analysis to identify bugs?