

# Clear Narratives

Designing and Maintaining Software (DAMS)

Louis Rose

# Your code tells a story

YEARS

# The Times-Picayune

nola.com

BREAKING NEWS AT **NOLA.COM**

WEDNESDAY, JUNE 13, 2012

METRO EDITION • 75¢

## Defense attorney deserts client midtrial

*North shore judge ships lawyer to jail*

By Claire Galofaro  
St. Tammany bureau

Two days into a St. Tammany Parish rape trial, the defense attorney stood and confessed to the judge that he'd done an inadequate job representing his client, a 25-year-old Slidell man accused of raping two young girls.

Attorney Claiborne Brown refused to proceed, despite the judge's threats to hold him in contempt of court.

Judge William Barris, with "righteous indignation," as he described his feelings on the matter, told the bailiffs to take

# PAPER LAYS OFF 200 EMPLOYEES

New companies to have 20 percent fewer workers



## DA cites progress while avoiding discord

*He beseeches mayor for more money*

By John Simmerman  
Staff writer

Orleans Parish District Attorney Leon Cannizzaro chose tact over attack in his annual address to the city Tuesday night, boasting progress for a notoriously dysfunctional New Orleans criminal justice system while plying Mayor Mitch Landrieu for more money.

Cannizzaro touted the successful prosecutions of some high-profile criminals, including last year's conviction of Uptown crime kingpin

# Confident Code

*“Code always tells a story (or tries to) but timid code [code with a poor narrative] often pauses and says ‘uhm’ and ‘ahh’”*

- Avdi Grimm

<https://www.youtube.com/watch?v=T8J0j2xJFgQ>



# Confident Narrative

*Do the following:*

*gather input  
perform work  
deliver results  
check for errors*

*in separate phases and in that order*

# Confident Narrative

```
def say(message, options={})
  command = "cowsay"
  if options[:strings] && options[:strings][:eyes]
    command << " -e '#{options[:strings][:eyes]}'"
  end

  messages = case message
              when Array then message
              when nil then []
              else [message]
              end

  results = []
  messages.each do |message|
    @io_class.popen(command, "w+") do |process|
      results << begin
        process.write(message)
        process.close_write
        result = process.read
      rescue Errno::EPIPE
        message
      end

      end
    end
  end
  output = results.join("\n")
  if options[:out]
    options[:out] << output
  end
  destination = case options[:out]
                when nil then "return value"
                when File then options[:out].path
                else options[:out].inspect
                end
  @logger.info "Wrote to #{destination}"
  if $? && ![0,172].include?($?.exitstatus)
    raise ArgumentError, "Command exited with status #{($?.exitstatus.to_s)}"
  end
  output
end
```

**Gather Input**  
**Perform Work**  
**Deliver Results**  
**Handle Errors**

# Confident Narrative

```
def say(message, options={})
  return "" if message.nil?
  options[:cowfile] and assert(options[:cowfile].to_s !~ /^\\s*$/)

  width      = options.fetch(:width) {40}
  eyes       = Maybe(options[:strings][:eyes])
  cowfile     = options[:cowfile]
  destination = WithPath.new(options[:out]).path
  out         = options.fetch(:out) { NilObject.new }
  messages    = Array(message)
  command     = "cowsay"
  command << " -W #{width}"
  command << " -e '#{options[:strings][:eyes}]'" unless eyes.nil?
  command << " -f #{options[:cowfile]}" unless cowfile.nil?

  results = messages.map { |message|
    checked_popen(command, "w+", lambda{message}) do |process|
      process.write(message)
      process.close_write
      process.read
    end
  }
  output = results.join("\\n")
  out << output

  @logger.info "Wrote to #{destination}"
  output
end
```

- Gather Input
- Perform Work
- Deliver Results
- Handle Errors

# Summary

Have in mind the reader of your code.  
A good story well told.

Avoid tangents and digressions by adopting a good narrative structure throughout your code.