When /bin/sh Strikes Back: Revisiting “Automate All The Things”

J. Paul Reed

NETFLIX

SRE CON Americas

December, 2020
Does Any of This Sound Familiar?

Invoice System

#SREcon  @jpauleyreed
Does Any of This Sound Familiar?

 Invoice System
Does Any of This Sound Familiar?

Invoice+Reporting System

#SREcon

@jpaulreed
Does Any of This Sound Familiar?

Invoice+Reporting+Payment System

#SREcon

@jpaulreed
Does Any of This Sound Familiar?

Invoice+Reporting+Payment System
Does Any of This Sound Familiar?

Automated Invoice+Reporting+Payment System

#SREcon @j paul reed
Does Any of This Sound Familiar?

“Automated” Invoice+Reporting+Payment System
Does Any of This Sound Familiar?

“Automated” & Monitored Invoice+Reporting+Payment System

#SREcon

@jpaulreed
Does Any of This Sound Familiar?

“Automated” & Monitored Invoice+Reporting+Payment System

#SREcon

@jpaulreed
Does Any of This Sound Familiar?

"Automated" & Monitored Invoice+Reporting+Payment System

#SREcon
Does Any of This Sound Familiar?

“Automated” & Monitored Invoice+Reporting+Payment System

#SREcon

@jpaulreed
Does Any of This Sound Familiar?

“Automated” & Monitored Invoice+Reporting+Payment System

#SREcon

@jpauleed
MINOR CHANGE

- NEW
- REWRITING
- SLIGHTLY CHANGED
- IMPACTED
- DO NOT TOUCH
- NOT OUR FAULT

WE NEED A FULL REGRESSION BY TOMORROW

MONKEYUSER.COM
Between Two Boeing Crashes,
Days of Silence and Mistrust
A Long “Drift Into Failure”

- Long running system, with legacy requirements
- *Eras* of modification
- Addition of supplemental features to address changes and maintain system *abstractions*
- Disparate, disconnected requirements gathering
- Siloed *testing* and validation
- Early, weak *stress signals* missed or ignored
$1,187,675
The 737Max and Why Software Engineers Might Want to Pay Attention

As someone with a bit of a reputation for talking about aviation and software development and operations, I've been asked about the 737Max...

link.medium.com

4:13 PM - 14 Mar 2019

652 Retweets 1,106 Likes

www.jpaulreed.com/737max-and-us
J. Paul Reed

- @jpaulreed on Twitter
- Alumn of The Ship Show
- 20+ Years in Build/Release Engineering
- Now, Critical Operations & Reliability Engineering team at Netflix
- Master of Science in Human Factors & System Safety
“We’re doing DevOps, OK?!”

#SREcon
@jpaulereed
“Universal” “Automation”

```
#!/bin/bash

pip install "$1" &
easy_install "$1" &
brew install "$1" &
npm install "$1" &
yum install "$1" & dnf install "$1" &
docker run "$1" &
pkg install "$1" &
apt-get install "$1" &
sudo apt-get install "$1" &
steamcmd +app_update "$1" validate &
git clone https://github.com/"$1"/$1 &
cd "$1"; ./configure; make; make install &
curl "$1" | bash &
```
A “Simple” Question?

What is “Automation?”
What is “Automation?”

“Script” — “Orchestration” — “ML/AI”
A “Simple” Question?

What is “Automation?”

“Runbooks”  “Script”  “Orchestration”  “ML/AI"
A “Simple” Question?

What is “Automation?”
A “Simple” Question?

What is “Automation?”

Dev

Ops

#SREcon

@jpaulreed
A World of Complex Automation

Should we send a notification?
A World of Complex Automation

#SREcon
Complex, *Interconnected* Automation

#SREcon

@jpaulreed
Complex, *Interconnected* Automation
Complex, Interconnected Automation
Complex, **Interconnected** Automation
AUTOMATE
ALL THE THINGS?
Ironies of Automation*

LISANNE BAINBRIDGE†

Key Words — Control engineering computer applications; man–machine systems; on-line operation; process control; system failure and recovery.

Abstract — This paper discusses the ways in which automation of industrial processes may expand rather than eliminate problems with the human operator. Some comments will be made on methods of alleviating these problems within the ‘classic’ approach of leaving the operator with responsibility for abnormal conditions, and on the potential for continued use of the human operator for on-line decision-making within human–computer collaboration.

Irony: combination of circumstances, the result of which is the direct opposite of what might be expected. Parody: seemingly absurd though perhaps really well-founded designer errors can be a major source of operating problems. Unfortunately people who have collected data on this are reluctant to publish them, as the actual figures are difficult to interpret. (Some types of error may be reported more readily than others, and there may be disagreement about their origin.) The second irony is that the designer who tries to eliminate the operator still leaves the operator to do the tasks which the designer cannot think how to automate. It is this approach which causes the problems to be discussed here, as it means that the operator can be left with an arbitrary collection of tasks, and little thought may have been given to providing support for them.

A Longstanding Question
Ironies of Automation*

LISANNE BAINBRIDGE†

Ironies: combination of circumstances, the result of which is the direct opposite of what might be expected. Paradox: seemingly absurd though perhaps really well-founded

This paper suggests that the increased interest in human factors among engineers reflects the irony that the more advanced a control system is, so the more crucial may be the contribution of the human operator. The increasing concern with human factors is in line with the observations of the ‘Irony’ that the more a control system is automated, the greater is the potential for unforeseen problems. (Some types of error may be reported more readily than others, and there may be disagreement about their origin.) The second irony is that the designer who tries to eliminate the operator still leaves the operator to do the tasks which the designer cannot think how to automate. It is this approach which causes the problems to be discussed here, as it means that the operator can be left with an arbitrary collection of tasks, and little thought may have been given to providing support for them.

† Lisanne Bainbridge holds a dissertation in architecture at the University of California and is currently a member of the faculty of the University of London. She has previously published in the field of human factors in systems.
Ironic Automation

- Manual skills deteriorate when they are not used.
- The generation of “new strategies” requires an adequate knowledge of the system.
- “There is some concern that the present generation of automated systems, which are monitored by former manual operators, are riding on their skills, which later generations of operators cannot be expected to have.”
- Automation generally requires a speed versus correctness tradeoff.
- Automation can camouflage current system state.
Ironic Automation, continued

- Automatic systems *should fail obviously*. 

#SREcon  @jpaulreed
YA THINK?
Ironic Automation, continued

- Automatic systems should fail obviously.
- Tracing the decision trees made by algorithms can be difficult (or today, impossible).
- This leads to the inability to fully understand the current context of the system when you are paged.
What is it doing now?!

How did I get into this mode?

How do I stop it from doing this?

What will it do next?

I know there is some way to get it to do what I want...

Stop interrupting me while I’m busy!

Why did it do this?
Ironic Automation, continued

- Automatic systems should fail obviously.
- Tracing the decision trees made by algorithms can be difficult (or impossible).
- This leads to the inability to fully understand the current context of the system when you are paged.
- “It is ironic to train operators to follow instructions, and then put them in the system to provide intelligence to it.”
Other Problematic Automation Aspects

- Automation often disconnected or treated distinctly from the application.
- Automation is not co-evolved with the application or treated as a product.
- Teams can (oddly?) devalue the implementation / ownership of automation.
# Joint Cognitive System

<table>
<thead>
<tr>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomy</td>
</tr>
<tr>
<td>Authority</td>
</tr>
<tr>
<td>Directed Attention</td>
</tr>
<tr>
<td>Redirectability</td>
</tr>
<tr>
<td>Interpredictability</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>Autonomy</td>
</tr>
<tr>
<td>Authority</td>
</tr>
<tr>
<td>Directed Attention</td>
</tr>
<tr>
<td>Redirectability</td>
</tr>
<tr>
<td>Interpredictability</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>Autonomy</td>
</tr>
<tr>
<td>Authority</td>
</tr>
<tr>
<td>Directed Attention</td>
</tr>
<tr>
<td>Redirectability</td>
</tr>
<tr>
<td>Interpredictability</td>
</tr>
</tbody>
</table>

Foundations of “Coordination”
Technologists often mistake connectivity (the technical capability to connect to disparate parties and data sources) for coordination. — Dave Woods
Automated systems, as they increase in autonomy and authority have two kinds of interpretations.
The “Animacy” Paradox

Automated systems, as they increase in autonomy and authority have two kinds of interpretations:

As a deterministic machine

#SREcon @jpaulereed
The “Animacy” Paradox

Automated systems, as they increase in autonomy and authority have two kinds of interpretations:

- As an animate agent capable of activities independent of the operator
- As a deterministic machine
As a deterministic machine

As an animate agent capable of activities independent of the operator

Automated systems, as they increase in autonomy and authority have two kinds of interpretations.

In Context!

In Hindsight!
The “Animacy” Paradox

Automated systems, as they increase in autonomy and authority have two kinds of interpretations

As an animate agent capable of activities independent of the operator

As a deterministic machine

In Context! In Hindsight!
The “Animacy” Paradox

Automated systems, as they increase in autonomy and authority, have two kinds of interpretations.

As an animate agent capable of activities independent of the operator.

As a deterministic machine.

In Context! In Hindsight!

#SREcon  @jpaulreed
The Rasmussen Triangle
The Rasmussen Triangle

- Boundary of Economic Failure
- Boundary of Unacceptable Workload
- Boundary of Functionally Acceptable Performance / Acceptable Risk

#SREcon @jpaulereed
The Rasmussen Triangle
The Rasmussen Triangle

"Cheaper, Better, Faster"

#SREcon @jpaulreed
The Rasmussen Triangle

“Cheaper, Better, Faster”

Maximum Work for Least Effort

#SREcon @jpaulreed
The Rasmussen Triangle

“Cheaper, Better, Faster”

Maximum Work for Least Effort

#SREcon

@jpaulreed
The Rasmussen Triangle

"Cheaper, Better, Faster"

Maximum Work for Least Effort

The "Discretionary Space"

#SREcon

@jpaulereed
The Rasmussen Triangle

The “Discretionary Space”

“Cheaper, Better, Faster”

Maximum Work for Least Effort
The Rasmussen Triangle

"Cheaper, Better, Faster"

Maximum Work for Least Effort

Automation Cannot Deal with The “Discretionary Space”

#SREcon @jpauleed
So... We Just Stop Automating Anything?!
So... We Just Stop Automating Anything?! (No.)
We’re obsessed in medicine with having great components—the best drugs, the best devices, the best specialists—but pay little attention to how to make them fit together well.

— Atul Gawande
The Checklist™

#SREcon  @jpaulereed
The Checklist™

#SREcon

@jpaulreed
## Pre-Release

### Checklist

<table>
<thead>
<tr>
<th>ID</th>
<th>Task</th>
<th>Dep Task</th>
<th>Exec</th>
<th>Vrfy</th>
<th>Notes/Deviations/Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Identify/confirm <strong>Release Reps</strong> from each team</td>
<td></td>
<td>P</td>
<td>All</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Identify/confirm <strong>projected release dates</strong></td>
<td></td>
<td>P</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Validate expected <strong>User Stories</strong> have landed</td>
<td></td>
<td>P</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Select <strong>Agent Version</strong> number</td>
<td></td>
<td>P</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>Create internal <strong>Release Summary</strong> document</td>
<td>1.3</td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>Determine whether this release will trigger an <strong>AFP run</strong></td>
<td>1.5</td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>Schedule <strong>QA</strong> Time block (9 hours); validate schedule with Engineering release team</td>
<td>1.2</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>Create release-specific <strong>Slack Channel</strong></td>
<td></td>
<td>E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Pre-Release

<table>
<thead>
<tr>
<th>ID</th>
<th>Task</th>
<th>Dep Task</th>
<th>Exec</th>
<th>Vrfy</th>
<th>Notes+Deviations+Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Identify/confirm <strong>Release Reps</strong> from each team</td>
<td></td>
<td>P</td>
<td>All</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Identify/confirm <strong>projected release dates</strong></td>
<td></td>
<td>P</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Validate <strong>expected User Stories</strong> have landed</td>
<td></td>
<td>P</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Select <strong>Agent Version</strong> number</td>
<td></td>
<td>P</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>Create internal <strong>Release Summary</strong> document</td>
<td>1.3</td>
<td>P</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>Determine whether this release will trigger an <strong>AFP run</strong></td>
<td>1.5</td>
<td>P</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>Schedule QA <strong>Time block</strong> (6 hours); validate schedule with Engineering release team</td>
<td>1.2</td>
<td>E</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>Create release-specific <strong>Slack Channel</strong></td>
<td>E</td>
<td></td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
The Real Value: A Requirements Spec

"I spend a lot of time on this task. I should write a program automating it!"

**Theory:**

- Writing Code
- Work on Original Task
- Automation Takes Over
- Free Time

**Reality:**

- Writing Code
- Debugging
- Rethinking
- Ongoing Development
- No Time for Original Task Anymore

Courtesy [XKCD](https://xkcd.com)

#SREcon @jpaulreed
We found that having branches or forks with very short lifetimes (less than a day) before being merged into trunk, and less than three active branches in total, are important aspects of continuous delivery, and all contribute to higher performance. So does merging code into trunk or master on a daily basis. Teams that don’t have code freeze periods (when people can’t merge code or pull requests) also achieve higher performance.

Overall, the technical practices that we found to be significant this year, along with their impact on culture and performance, are shown below in Figure 1. Findings that are new this year are shown in bold.

- Lower change fail rates
- Comprehensive, fast and reliable test and deployment automation
- Trunk-based development and continuous integration
- Effective test data management
- Application code and app & system configuration all in version control
- Incorporating security (and security teams) into the delivery process

Together, the factors on the left model continuous delivery, which leads to...

- Less rework
- Lower levels of deployment pain
- Higher levels of IT performance (higher throughput and stability)
- Identifying strongly with the organization you work for
- Lower change fail rates
- Higher levels of org performance (productivity, market share, profitability)

**2016 State of DevOps Report**

#SREcon
Version Control: Mattering. Still.

And yet...
Dealing With the Ironies

- Engage in practices that cultivate ability to “buy time”
  - Simulation (i.e. Chaos Engineering, game days, etc.)
  - Widen system understanding
  - “Checks” over “locks”
- Consider the element of time pressures when designing automation
Framing Automation Coherently

- Chef / Puppet / Ansible / etc.
- **QuickRelease**
  - Execute and Verify
  - Rerun single steps
  - Automatic logging
## Pre-Release Checklist

<table>
<thead>
<tr>
<th>ID</th>
<th>Task</th>
<th>Dep Task</th>
<th>Exec</th>
<th>Vrfy</th>
<th>Notes + Deviations + Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Identify/confirm <strong>Release Reps</strong> from each team</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Identify/confirm <strong>projected release dates</strong></td>
<td>P</td>
<td></td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Validate <strong>expected User Stories</strong> have landed</td>
<td>P</td>
<td></td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Select <strong>Agent Version</strong> number</td>
<td>P</td>
<td></td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>Create internal <strong>Release Summary</strong> document</td>
<td>1.3</td>
<td>P</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>Determine whether this release will trigger an <strong>AFP run</strong></td>
<td>1.5</td>
<td>P</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>Schedule <strong>QA</strong> Time block (6 hours); validate schedule with Engineering release team</td>
<td>1.2</td>
<td>E</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>Create release-specific <strong>Slack Channel</strong></td>
<td>E</td>
<td></td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Framing Automation Coherently

- Chef / Puppet / Ansible / etc.
- QuickRelease
  - Execute and Verify
  - Rerun single steps
  - Automatic logging
- Rerun
```
$ ./rerun --help

rerun
Version: v0.1. License: Apache 2.0.

Usage: rerun [-h][--version] [-M <dir>] [module:command [options]]

Examples:
$ rerun
  => List all modules.
$ rerun waitfor
  => List all waitfor commands.
$ rerun waitfor:http --url http://google.com
  => Execute the waitfor:http command.
$ rerun -M /var/rerun/waitfor:http
  => Execute the waitfor:http command found in /var/rerun

Listing
Without arguments, rerun will list existing modules;
```
## Improving Coordination

<table>
<thead>
<tr>
<th></th>
<th>Operators</th>
<th>Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomy</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Authority</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Directed Attention</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Redirectability</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Interpredictability</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

---

# SREcon

@jpaulreed
## Improving Coordination

<table>
<thead>
<tr>
<th></th>
<th>Operators</th>
<th>Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomy</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Authority</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Limited Control Span</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Decomposable Stop/Start</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Interpredictability</td>
<td>✔️</td>
<td>😞</td>
</tr>
</tbody>
</table>
# Improving Coordination

<table>
<thead>
<tr>
<th></th>
<th>Operators</th>
<th>Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomy</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Authority</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Limited Control Span</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Decomposable Stop/Start</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Interpredictability</td>
<td>✔️</td>
<td>😕</td>
</tr>
<tr>
<td>Retrospective Audit</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

#SREcon  
@jpaulreed
So... We Can Keep Automating Things??

(Yes!)
Automation Isn’t “Dangerous” (usually) but the way Our Industry Thinks About It… Can Be.
Automation Must Be Designed.

(And That’s a Team Sport.)
Automation Should Be Treated Like a Product (With an Owner)
Automation That Truly Participates in Our Cognitive Joint Systems Is Still Nacent
Go forth... and Automate!

J. Paul Reed
www.jpaulreed.com
@jpaulreed