L09 Configuration as a User Interface

Markus Raab

Institute of Information Systems Engineering, TU Wien

09.06.2021

3-way merge

Learning Outcomes

Students will be able to

- recall a method of avoiding errors
- apply some principles of good error messages
- remind some basics of system administrator research

Synchronization

Problem: transient and persistent configuration settings might be out-of-sync [7]

Requirement

Configuration libraries must provide ways to keep transient and persistent views consistent.

Solutions:

- Often write out configuration settings.

Semantic 3-way merge

Problem: When trying to writing out configuration settings, the configuration settings might not be as they were before. (Conflict)
Solution: Many conflicts can be resolved automatically with a semantic 3-way merge.
We can resolve many conflicts automatically if we consider:

- the key/value structure (vs. line-based)
- the origin of the configuration settings
- the type of settings

For example, when upgrading slapd:

- System administrator changed the file (Ours).
- Package maintainer changed the file (Theirs).

Conflicts Example

Ours:

```
1 slapd/threads/listener=4
2
3 slapd/threads/enable= \
4     yes # must be enabled for listener
5
```

Theirs:

```
1 slapd/threads/enable = on
2 slapd/threads/listener = 8
```

Origin:

```
1 slapd/threads/listener=8
2 slapd/threads/enable = true
```

L09 Configuration as a User Interface

Markus Raab

Institute of Information Systems Engineering, TU Wien

09.06.2021

Error Messages

Motivation (Recapitulation)

Error messages are extremely important as they are the main communication channel to system administrators.

```
1 [a]
2   check/type:=long
3 [b]
4   check/type:=long
5 [c]
6   check/range:=0-10
7   assign/math:=../a+../b
```

Task

Where should the error message point to if we change b to 10 (a is unchanged 1)?

Considerations (Recapitulation)

Task

What needs to be considered when designing error messages?

- Generic vs. specific plugins
- Precisely locate the cause (and do not report aftereffects)
- Give context
- Personification [8]

Further Considerations

- configuration design first: avoid errors if possible
- "edit here mentality": do not point to correct statements [9]
- precision and recall[1] [11]
- error messages should not leak internals [4]
- do not propose solutions [9] if you are not sure
- reduce vocabulary [9]
- tension between providing enough information and not overwhelming the user [11]
- colors might help [11]

[1] terms from classification, it is the numerical counterpart of soundness and completeness

Error Messages for Misconfiguration [12]

- error messages are often the sole data source
- tool uses misconfiguration injection and checks if error message point to the correct setting
- tool requires system tests
- they considered error message as okay if key *or* value is present

Implication

Missing error message means the configuration specification is not complete.

Context for error messages

Error messages should contain:

- pin-point key (which also pin-points to the specification)
- repeat relevant parts of values and the specification
- show mountpoint (to make relative keys unique)
- show file name and line number
- for reporting bugs: show source code lines

Precise Location (Recapitulation)

```
1 a=5   ; unmodified
2 b=10  ; modification bit in metadata
3       ; is only set here
4 c=15  ; unmodified by user but changed
5       ; later by assign/math
```

Example Error Messages (Recapitulation)

```
Sorry, I was unable to change the configuration settings!
Description: I tried to set a value outside the range!
Reason: I tried to modify b to be 10 but this caused c to
        be outside of the allowed range (0-10).
Module: range
At: sourcefile.c:1234
Mountpoint: /test
Configfile: /etc/testfile.conf
```

Example Error Messages (Improvement)

```
Sorry, module range issued error C03100:
I tried to modify b to be 10 but this caused c to
be outside of the allowed range (0-10).
```

L09 Configuration as a User Interface

Markus Raab

Institute of Information Systems Engineering, TU Wien

09.06.2021

System Administrator Research

User View

Who is the user of CM?

- End Users?
- Developers (devs)?
- System Administrators (admins)?

System Administrator Research

- Interest of understanding administrators emerged around 2002 [1].
- Typical methods are surveys, diary studies, interviews and observations (ethnographic field studies).
- Field studies also done in industry [3].
- Barrett [2] tried to initiate a workshop at CHI 2003 to draw the attention of the HCI community towards system administration.
- The workshop was already dropped in the next year.
- The tenor is that "tools ... are not well aligned" [6].
- Research mainly looks at pre-CM. Manual administration is still standard (Source: e.g., Luke Kanies).

CM research

In the meanwhile at Large Installation System Administrator Conference (LISA):

- began as CFengine Workshop at LISA 2001
- CM workshop by Paul Anderson [1]
- in LISA 2003 an informal poll asked about CM tools:
  the only user of each tool in the room at the time was its author [5]
- it is easy to invent CM tools (and configuration file formats)
- it is difficult to make it useful beyond your own goals

Tasks

What do system administrators do?

- keep our infrastructure running
- coordinate
- do backups
- manage hardware
- do inventory
- install applications
- manage security
- configure applications
- troubleshoot
- $\Longrightarrow$ the unsung heroes!

7 people, 1 command-line [3]

- system administrator misunderstood problem
  (had a wrong assumption)
- 7 people sought attention and trust, competing to tell the admin what to do
- due to wrong assumption the admin communicated to everyone, people could not help
- there were several instances in which the admin ignored or misinterpreted evidence of the real problem
- eventually someone else solved the problem: admin confused “from”/“to” port in the settings and firewall blocked requests

other cases [3]

- lost semicolon: execution of script failed due to missing semicolon, then they tried to delete a non-existent table.
- crontab: onltape/ofltape confused because of discussion about offline backup (although an online backup should be performed).
- crit sit: many system administrators competed against each other trying to write a simple script. The crit sit continued for two weeks.

Haber and Bailey [6]

Later Haber and Bailey [6] repeated an ethnographic field study. The stories are similar to Barrett et al. [3]. Their study was also conducted in the same company. They created personas:

- database administrator
- web administrator
- security administrator

Database Administrator [6]

- frequent contact via phone, e-mail and IM
- needs to work on weekends
- pair-programming for new tasks
- typical errors: stopping wrong database process

Web Administrator [6]

- crit sit
- deploying new Web applications
- about 20-400 steps to deploy an application
- moving from test to production done by hand

Security Administrator [6]

- gets emails on suspicious activities
- multi-user chat
- ad-hoc scripts

Haber and Bailey [6]

- “if data is lost...that is when you write your résumé.”
- 90 % is spent with communicating with other admins
- only 6 % is gathering information and running commands
- quality control: monitoring found that non-functional service was down two days

Barrett et al. [3]

- 20 % of the time is spent in diversions
- 20 % of the time people communicated about *how to communicate*
- CLIs were generally preferred
- configuration and log files are scattered, poorly organized and often used inconsistent terminology

Findings [3]

- syntax checking is essential
- replicating actions (e.g., to production) is error-prone
- undo not available
- do not assume a complete mental model ("if understand the system is a prerequisite [...], we are lost")
- do not assume programming skills (only 35 % reported having a bachelor's degree)
- trust in CLI tools but little trust in GUIs (is the information up-to-date?)
- errors while executing scripts lead to inconsistent state, rerunning often does not work

  (not idempotent)

Design Principles [6]

Many design principles for tools were given [6]:

- configuration and logs should be displayed in a uniform way
- APIs/plugins for tools should be provided
- errors in configuration need to be discovered quickly
- confusion of similar settings should be avoided
- provide means of comparing configuration settings
- provide consistent profiles of information
- both transient and persistent settings should be visible
- when errors occur: always display which changes have been made (modern approach is idempotence)

Apply to CM

What can we learn from manual system administration?

- \+ intensive review process catches errors
- – collaboration ineffective
- – context/situational awareness is essential
- \+ precise editing of configuration files works well
- \+ self-written tools are very efficient

Idea

Replicate parts that work well, automate error-prone parts.

Precise Editing

Partial modifications (precise editing) is natural for humans.
It ensures preservations of (potentially security-relevant!) defaults.
In CM following methods are used:

- embed shell commands to do the work
- replace full content of configuration files
- replace full content of configuration files with templates
- line based manipulation (e.g., file_line): match line and replace it
- Augeas/XML: match a key with XPath and replace it
- Elektra: set the value of a key

Apply to CM

Elektra's goals are:

- it should be easy to develop new high-level tools
- precise editing: change the configuration value as specified

Administrators/Devs still need to:

- intensively review and improve the specifications
- test (and debug) configuration settings

Open topics (incomplete):

- safe migrations of settings and data
- collaboration
- management (including knowledge)

Conclusion

- Configuration management languages differ widely.
- Configuration specifications are helpful in different ways.
- Do not design around tools but design tools around you.

L09 Configuration as a User Interface

Markus Raab

Institute of Information Systems Engineering, TU Wien

09.06.2021

Meeting

Learning Outcomes

Students will be able to

- recall a method of avoiding errors
- apply some principles of good error messages
- remind some basics of system administrator research

Conflicts Example

How can you avoid errors when merging these configuration settings?

Ours:

```
1 slapd/threads/listener=4
2
3 slapd/threads/enable= \
4     yes # must be enabled for listener
5
```

Theirs:

```
1 slapd/threads/enable = on
2 slapd/threads/listener = 8
```

Origin:

```
1 slapd/threads/listener=8
2 slapd/threads/enable = true
```

Context for error messages

Error messages should contain:

- pin-point key (which also pin-points to the specification)
- repeat relevant parts of values and the specification
- show mountpoint (to make relative keys unique)
- show file name and line number
- for reporting bugs: show source code lines

Task

Break.

Tasks

What do system administrators do?

- keep our infrastructure running
- coordinate
- do backups
- manage hardware
- do inventory
- install applications
- manage security
- configure applications
- troubleshoot
- $\Longrightarrow$ the unsung heroes!

Task

Tell a story about system administrators.

Finalize all PRs:

- Rebase PRs (+1P).
- Make CI happy.
- Correction for H3 (for JNI: only with tests, no mounting).
- Correction for T3.

Feedback

- ECTS breakdown realistic?
- Feedback Talk
- TISS Feedback from 16.06.2021 00:00 to 14.07.2021 23:59

Outlook

How “everything” in CM connects:

- documentation
- introspection
- (code) generation
- context awareness

Best topics at last.

[1] Eric Arnold Anderson. *Researching system administration*. PhD thesis, University of California at Berkeley, 2002.

[2] Rob Barrett, Yen-Yang Michael Chen, and Paul P. Maglio. System administrators are users, too: Designing workspaces for managing internet-scale systems. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pages 1068–1069, New York, NY, USA, 2003. ACM. ISBN 1-58113-637-4. doi: 10.1145/765891.766152. URL http://dx.doi.org/10.1145/765891.766152.

[3] Rob Barrett, Eser Kandogan, Paul P. Maglio, Eben M. Haber, Leila A. Takayama, and Madhu Prabaker. Field studies of computer system administrators: analysis of system management tools and practices. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 388–395. ACM, 2004.

[4] P. J. Brown. Error messages: The neglected area of the man/machine interface. *Commun. ACM*, 26(4):246–249, April 1983. ISSN 0001-0782. doi: 10.1145/2163.358083. URL http://doi.acm.org/10.1145/2163.358083.

[5] Mark Burgess and Alva L Couch. Modeling next generation configuration management tools. In *LISA*, pages 131–147, 2006.

[6] Eben M. Haber and John Bailey. Design guidelines for system administration tools developed through ethnographic field studies. In *Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology*, CHIMIT '07, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-635-6. doi: 10.1145/1234772.1234774. URL http://dx.doi.org/10.1145/1234772.1234774.

[7] Dongpu Jin, Xiao Qu, Myra B. Cohen, and Brian Robinson. Configurations everywhere: Implications for testing and debugging in practice. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 215–224, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2768-8. doi: 10.1145/2591062.2591191. URL http://dx.doi.org/10.1145/2591062.2591191.

[8] Michael J. Lee and Andrew J. Ko. Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the Seventh International Workshop on Computing Education Research*, ICER '11, pages 109–116, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0829-8. doi: 10.1145/2016911.2016934. URL `http://dx.doi.org/10.1145/2016911.2016934`.

[9] Guillaume Marceau, Kathi Fisler, and Shriram Krishnamurthi. Mind your language: On novices' interactions with error messages. In *Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2011, pages 3–18, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0941-7. doi: 10.1145/2048237.2048241. URL `http://doi.acm.org/10.1145/2048237.2048241`.

[10] Markus Raab and Gergö Barany. Introducing context awareness in unmodified, context-unaware software. In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE,*, pages 218–225. INSTICC, ScitePress, 2017. ISBN 978-989-758-250-9. doi: 10.5220/0006326602180225.

[11] John Wrenn and Shriram Krishnamurthi. Error messages are classifiers: A process to design and evaluate error messages. In *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2017, pages 134–147, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5530-8. doi: 10.1145/3133850.3133862. URL `http://doi.acm.org/10.1145/3133850.3133862`.

[12] Sai Zhang and Michael D. Ernst. Proactive detection of inadequate diagnostic messages for software configuration errors. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, ISSTA 2015, pages 12–23, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3620-8. doi: 10.1145/2771783.2771817. URL `http://dx.doi.org/10.1145/2771783.2771817`.