

Configuration Management

Markus Raab

Institute of Information Systems Engineering, TU Wien

20.3.2018



Lecture is every week Wednesday 09:00 - 11:00.

06.03.2019: topic, teams

13.03.2019: TISS registration, initial PR

20.03.2019: other registrations, guest lecture

27.03.2019: first issue done, second assigned,
HS: kleiner Schiffbau

03.04.2019:

10.04.2019: mid-term submission of exercises

08.05.2019: (HS?)

15.05.2019:

22.05.2019:

29.05.2019:

05.06.2019: final submission of exercises

12.06.2019:

19.06.2019: last corrections of exercises

26.06.2019: exam

Popular Topics

14	tools	4	design
9	testability	4	cascading
9	code-generation	4	architecture of access
7	context-awareness	3	configuration sources
6	specification	3	config-less systems
6	misconfiguration	2	secure conf
6	complexity reduction	2	architectural decisions
5	validation	1	push vs. pull
5	points in time	1	infrastructure as code
5	error messages	1	full vs. partial
5	auto-detection	1	convention over conf
4	user interface	1	CI/CD
4	introspection	0	documentation

Talk

about anything related to configuration management.

- The duration must be not longer than 20 minutes (shorter is ok, content matters).
- It must be about your experience.
 - E.g., about the homework you did.
 - I.e., not only about study of literature.
 - If you extensively use some tool before, please share your experience.
- Two persons per date.
- Same topic allowed if persons coordinate their talk.
- **Slides must be in repo**
(please CC licensed and with source)

Tasks for today

(until 20.03.2019 23:59)

Task

Make sure to say which programming languages you know in STUDENTS.ini.

Task

Registration for talk, homework, teamwork.

Task

Write text in at least one issue.

Tasks for next week

(until 27.03.2019 23:59)

(hint: to get help submit at least one day earlier)

Task

Description of homework.

Task

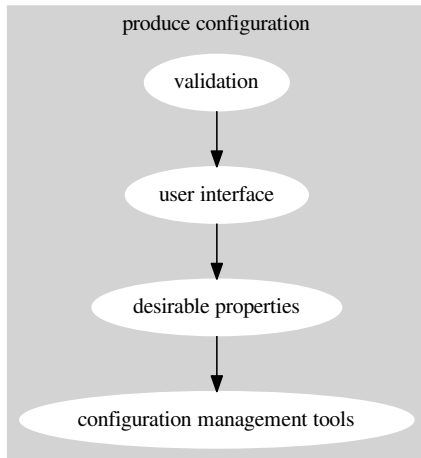
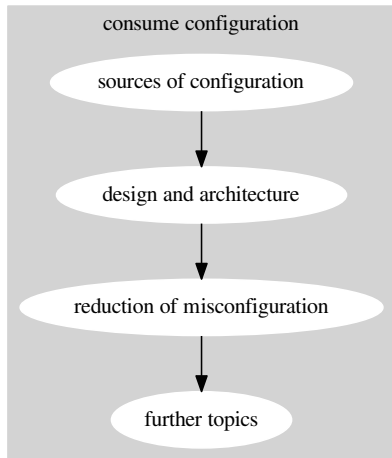
Description of teamwork (which application, which CM tool).

Either:

- Inside a folder (use GitHub name for the folder name) of the private repo as pull request.
- As new issue in the public libelektra repo.

Task

Fix at least one issue and write some text in at least one other issue.



Command-line Arguments

1 Command-line Arguments

- Usage and Popularity
- Semantics

2 Environment Variables

- Requirements
- Conclusion

3 Complexity

- Trend
- Calculation
- Usage

Is there something else?

- configuration files are the most researched of all configuration sources [3]
- but it is neither the most used nor most popular [5]

Q: “Which configuration systems/libraries/APIs have you already used or would like to use in one of your FLOSS project(s)?”

- command-line arguments (92 %, $n = 222$)
- environment variables (79 %, $n = 218$)
- S: API `getenv` is used omnipresently with 2,683 occurrences
- configuration files (74 %, $n = 218$))

Q: *“What is your experience with the following configuration systems/libraries/APIs?”*

- getenv (10 %, $n = 198$)
- configuration files (6 %, $n = 190$)
- command-line options (4 %, $n = 210$)
- X/Q/GSettings (41 %, 14 %, 35 %)
- KConfig (21 %)
- dconf (42 %)
- plist (32 %)
- Windows Registry (69 %)

Task

Which configuration source do you use most?

Task

Possible talk: About one of these sources.

- passed by main for a new process via
(int argc, char ** argv)
- visible from other processes (e.g., via ps aux)
- could be passed along to subprocesses but hardly done
- need to be parsed by process
- portability: differences in parsing
- cannot be changed from outside (requires restart, no IPC)

Environment Variables

1 Command-line Arguments

- Usage and Popularity
- Semantics

2 Environment Variables

- Requirements
- Conclusion

3 Complexity

- Trend
- Calculation
- Usage

Semantics

- are also per-process (`/proc/self/environ`)
- are not visible from other processes
- are automatically inherited by subprocesses
- need to be parsed by process (`[extern] char **environ`)
but API is provided (`getenv`)
- cannot be changed from outside (requires restart or an additional IPC mechanism)

Task

What is wrong with the code in the book?

getenv

- is widely standardized, including SVr4, POSIX.1-2001, 4.3BSD, C89, C99 [1],
- is supported by many programming languages, and
- enforces key=value convention.

Usage

- ① bypassing other configuration accesses (Q: 45 %)
- ② locating configuration files
- ③ debugging and testing (Q: 55 %, S: 1,152, i. e. 43 %)
- ④ sharing configuration settings across applications (Q: 53 %, S: 716, i. e. 47 %)
- ⑤ for configuration settings unlikely to be changed by a user (Q: 20 %)
- ⑥ *“even when it is used inside a loop”* (Q: 2 %)

Portability

- no separators for values defined
- case sensitivity problems
- often many environment variables for the same purpose: TMP, TEMP, or TMPDIR
- sometimes one environment variable for different purposes: PATH
- if applications directly access `environ` multi-threaded applications have undefined behavior

How can we avoid the problems of environ?

Requirement

The configuration data structure should not be directly accessible but an API should be provided.

In Elektra KeySet fulfils the requirement.

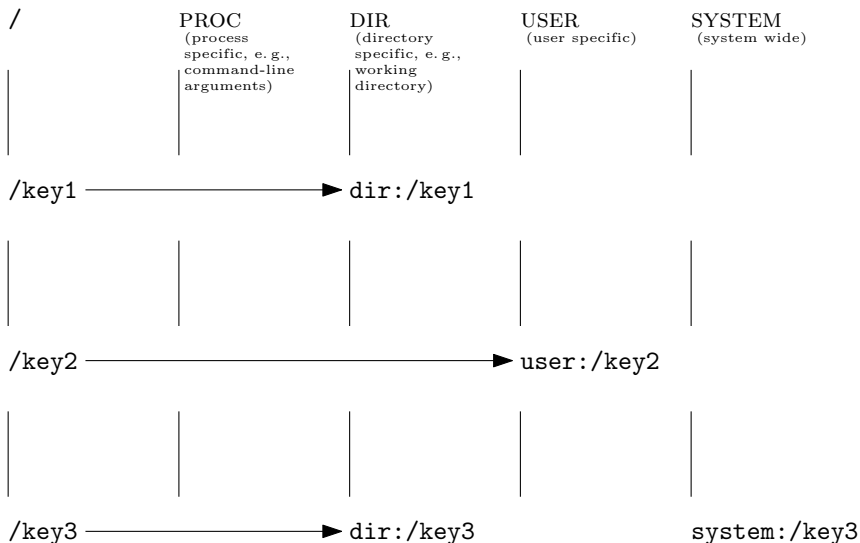
How can we deal with the many sources?

Requirement

A configuration library must support all three popular ways for configuration access: configuration files, command-line options, and environment variables.

In Elektra cascading of namespaces fulfils the requirement.

Cascading



Task

Discuss the differences of mounting and cascading with your neighbor.

User View

- command-line for trying out configuration settings
- environment variables for configuration settings within a shell
- configuration files for persistent configuration settings

Conclusion

- three different configuration sources widely used
- all three used for different reasons but often for the same configuration settings
- many different configuration file formats
- abstractions: key-value, mounting, and cascading

Complexity

1 Command-line Arguments

- Usage and Popularity
- Semantics

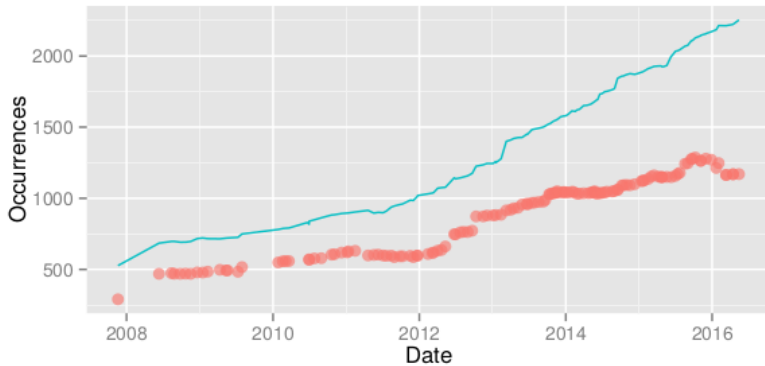
2 Environment Variables

- Requirements
- Conclusion

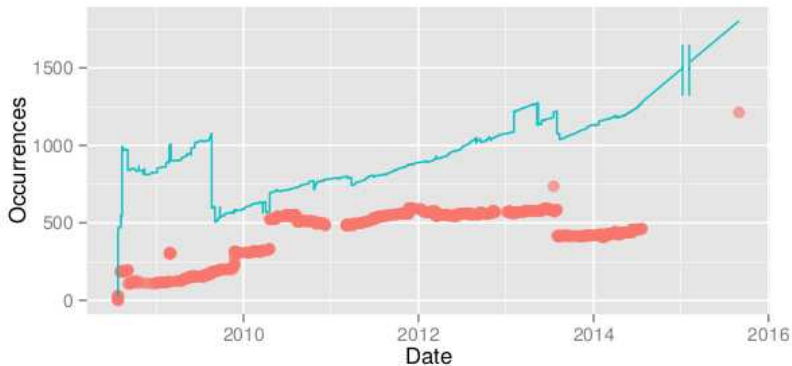
3 Complexity

- Trend
- Calculation
- Usage

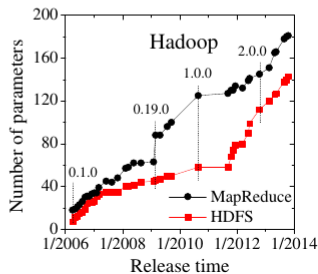
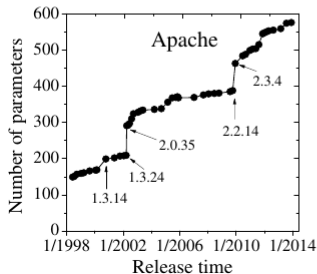
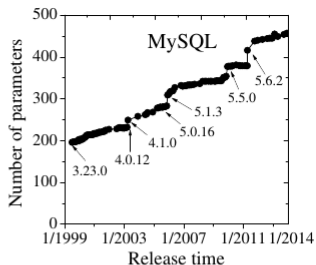
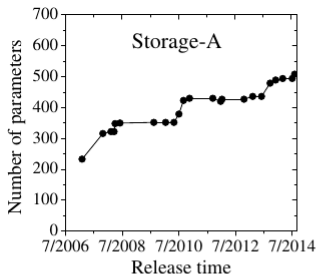
Trend Firefox



Trend Chromium



Trend Configuration Files



Types of Complexity

- complexity in access:
 - many different formats
 - non-uniformity
 - transformations
- configuration settings
 - number of settings s
 - number of values n
 - dependences between settings

Calculation of Complexity

Using enumerative combinatorics:

- number of configurations: n^s
- for N groups of different n and s (i.e., $n_1 \dots n_N$ with $s_1 \dots s_N$ occurrences):

$$\prod_{i=1}^N n_i^{s_i}$$

- more difficult to calculate (or unbounded) for dependences, module instantiations, arrays, ...

Calculation of Complexity

Examples:

- 600 boolean settings in Apache httpd (let us assume $n = 2$):
 $2^{600} \approx 10^{180}$
- 19 integer settings: $2^{32^{19}} = 2^{32 \cdot 19} = 2^{609} \approx 10^{183}$
- 2000 boolean settings in Firefox [3]: $2^{2000} \approx 10^{602}$

Calculation of Complexity (cont.)

Examples:

- for 20 boolean and 20 enums with 5 possibilities:

$$2^{20} * 5^{20} = 10^{20}$$

- MySQL has 461 settings, of which 216 are non-simple types [8]
(let us assume $n = \{3, 20\}$): $3^{245} * 20^{216} \approx 10^{397}$
(settings are explained in 5560 pages¹)
- an array with 1 – 20 boolean settings: 2^{20}

¹<https://downloads.mysql.com/docs/refman-5.7-en.pdf>

Task

Calculate complexity for some tool you know.

Task

Possible Homework: Write tool to calculate complexity with a given configuration specification.

Decision Tree

- configuration settings may depend on each other
- form a decision tree [2, 7]
- the decision tree is an instantiation of chosen configuration settings
- calculation only needs to consider instantiations which make a difference:
essential configuration complexity [4]

Harmful Defaults [8]

- Problem: Two major data losses on a dozen machines.
- Cause: Stayed with the default values of the data-path settings (e.g., `dfs.name.dir`, `dfs.data.dir`) which point to locations in `/tmp`. Thus, after the machines reboot, data losses occur. “One of the common problems from users.” (from Cloudera)
- up to 53 % of misconfigurations is due to staying at defaults
- 17 % to 48 % of configuration issues are about difficulties in finding settings

Unnecessary Settings [8]

- Configuration Parameter:
`dfs.namenode.tolerate.heartbeat.multiplier`
- Developers' Discussion: Since we are not sure what is a good choice, how about making it configurable? We should add a configuration option for it. Even if it's unlikely to change, if someone does want to change it they'll thank us that they don't have to change the code/recompile to do so.
- Real-World Usage:
 - No usage found by searching the entire mailing lists and Google.
 - No usage reported in a survey of 15 Hadoop users in UCSD.

Unnecessary Settings [8]

- 6 % to 17 % of settings set by majority
- up to 54 % are seldom set
- up to 47 % of numeric settings have no more than five distinct values

Reduction

Q: *"Why do you think configuration should be reduced?"*

- to simplify code maintenance (50 %),
- to prevent errors and misconfiguration (43 %),
- to provide better user experience (40 %),
- ***"I do not think it should be reduced"*** (30 %),
- because they prefer auto-detection (29 %)
(with a possibility to override configuration settings: 32 %),
- *"because use-cases which are rarely used should not be supported"* (13 %),
- *"never find time for this task"* (9 %), and
- *"because only standard use-cases should be supported"* (1 %)

Question

How to specify reduction strategies of configuration settings?

Answer

Configuration Specification

- [1] *getenv(3) Linux User's Manual*, March 2017.
- [2] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. Cool features and tough decisions: A comparison of variability modeling approaches. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, VaMoS '12*, pages 173–182, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1058-1. doi: 10.1145/2110147.2110167. URL <http://dx.doi.org/10.1145/2110147.2110167>.
- [3] 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>.

- [4] J. Meinicke, C. P. Wong, C. Kästner, T. Thüm, and G. Saake. On essential configuration complexity: Measuring interactions in highly-configurable systems. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 483–494, Sept 2016.
- [5] Markus Raab and Gergö Barany. *Challenges in Validating FLOSS Configuration*, pages 101–114. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57735-7. doi: 10.1007/978-3-319-57735-7_11. URL http://dx.doi.org/10.1007/978-3-319-57735-7_11.

- [6] 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.
- [7] Mark-Oliver Reiser. *Core Concepts of the Compositional Variability Management Framework (CVM): A Practitioner's Guide*. TU, Professoren der Fak. IV, 2009.
- [8] Tianyin Xu, Long Jin, Xuepeng Fan, Yuanyuan Zhou, Shankar Pasupathy, and Rukma Talwadker. Hey, you have given me too many knobs! Understanding and dealing with over-designed configuration in system software. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015*, pages 307–319, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3675-8. doi:

10.1145/2786805.2786852. URL
[http://dx.doi.org/10.1145/2786805.2786852.](http://dx.doi.org/10.1145/2786805.2786852)