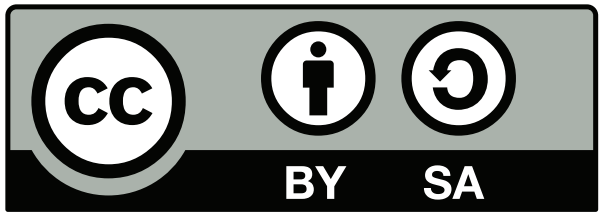
Configuration Management

Markus Raab

Institute of Information Systems Engineering, TU Wien

13.3.2018



Language of the Talk?

Task

Hands up if you prefer German.

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: (HS?)
03.04.2019:
08.05.2019: (HS?)
10.04.2019: mid-term submission of exercises
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
9	testability
9	code-generation
7	context-awareness
6	specification
6	misconfiguration
6	complexity reduction
5	validation
5	points in time
5	error messages
5	auto-detection
4	user interface
4	introspection
4	design
4	cascading
4	architecture of access
3	configuration sources
3	config-less systems
2	secure conf
2	architectural decisions
1	push vs. pull
1	infrastructure as code
1	full vs. partial
1	convention over conf
1	CI/CD
0	documentation

Organisation

- https://github.com/orgs/ElektraInitiative/teams/cm2019s now created
- private repo created
- first PRs created and accepted:
  - release notes often not updated
  - please set "ready to merge" if build server is happy

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.

Next tasks

Task

Initial PR must be done **today**, can be trivial

Task

Registration for talk, homework, teamwork until 20.03.2019 23:59

Task

Assign yourself at least one issue or ask me to assign one to you.
(make sure to say which programming languages you know)
Deadline: 20.03.2019 23:59

learning outcome:

- Having an overview of configuration file formats.
- Understanding Elektra's abstractions that deals with the multitude of configuration file formats.

Configuration File Formats

Basic Definitions

The ***execution environment*** is information outside the boundaries of each currently running process [5].
Controlling the execution environment is essential for configuration management [4, 10], testing [24, 28], and security [8, 13, 16, 22].

Configuration Setting

Definition

A ***configuration setting***, or ***setting*** in short, fulfills these properties:

1. It is provided by the execution environment.
2. It is *consumed* by an application.
3. It consists of a key, a configuration value, and potentially *metadata*. The ***configuration value***, or ***value*** in short, influences the application's behavior.
4. It can be *produced* by the maintainer, user, or system administrator of the software.

Synonyms for Configuration Settings

User preferences [11] and ***customization*** [1] stress that users make the change although that might not always be the case. ***Variability points*** [9, 14, 15, 25–27] aim at describing the capability of software to adapt its behavior. ***Derivation decision*** [6, 7] puts the decisions to make and not the result in focus. ***Configuration parameter*** [2, 30] is easily confused with other kinds of parameters. ***Configuration item*** [3] or ***configuration option*** [21, 31, 32] are sometimes not applicable, for example, “proxy option”, or “language item”. ***Configuration data*** [10] is often used in the context of programmable gate arrays and has a different meaning in that domain.

Definition

A ***configuration file*** is a file containing configuration settings.
A Web server configuration file:

```
1 port=80 ; comment
2 address=127.0.0.1
```

Task

What are keys? What are configuration values? What is metadata?

The configuration values are 80 and 127.0.0.1, respectively.
Other information in the configuration file is metadata for the configuration settings (such as the comment).

Types of Formats

- CSV (comma-separated values)
- semi-structured
- programming language
- document-oriented
- literate

CSV formats

- passwd: 3rd November, 1971
- passwd and group use : as separator
- are difficult to extend (e.g., GECOS)
- today mostly used for legacy reasons
- are replaced one-by-one (e.g., inetd, crontab)

Trends

- away from CSV
- towards general-purpose serialization formats (INI, JSON)
- human-read/writable (YAML, HOCON, TOML)
- programming language as configuration file

Programming Language

+ very easy for developers (simply source the file)
+ above-overage quality of error message
– makes automatic change of individual values harder
– very hard to use for people who do not know the programming language
– does not separate code and data

Introduce somebody

Task

Talk with someone about your favourite configuration file format.

Task

Did you implement a configuration file parser and/or invented a new configuration file format?

Task

Explain to everyone about the other person and his/her favourite configuration file format.

Method

What do FLOSS developers say?

- *Q:* survey with 672 persons visiting, 162 persons completing the survey [18]
- *S:* source code analysis of 16 applications, comprising 50 million lines of code [18]

Why are so many formats present?

Q: "In which way have you used or contributed to the configuration system/library/API in your previously mentioned FLOSS project(s)?" [18]

- 19 % persons ($n = 251$) have introduced a configuration file format.
- 29 % implemented a configuration file parser.
- 15 % introduced a configuration system/library/API.
- 34 % used external configuration access APIs.

Multitude of Formats

- on every system a multitude of (legacy) configuration file formats exist
- the number grows fast
- thus applications usually have to deal with some legacy formats

Requirement

A configuration library must be able to integrate (legacy) systems and must fully support (legacy) configuration files.

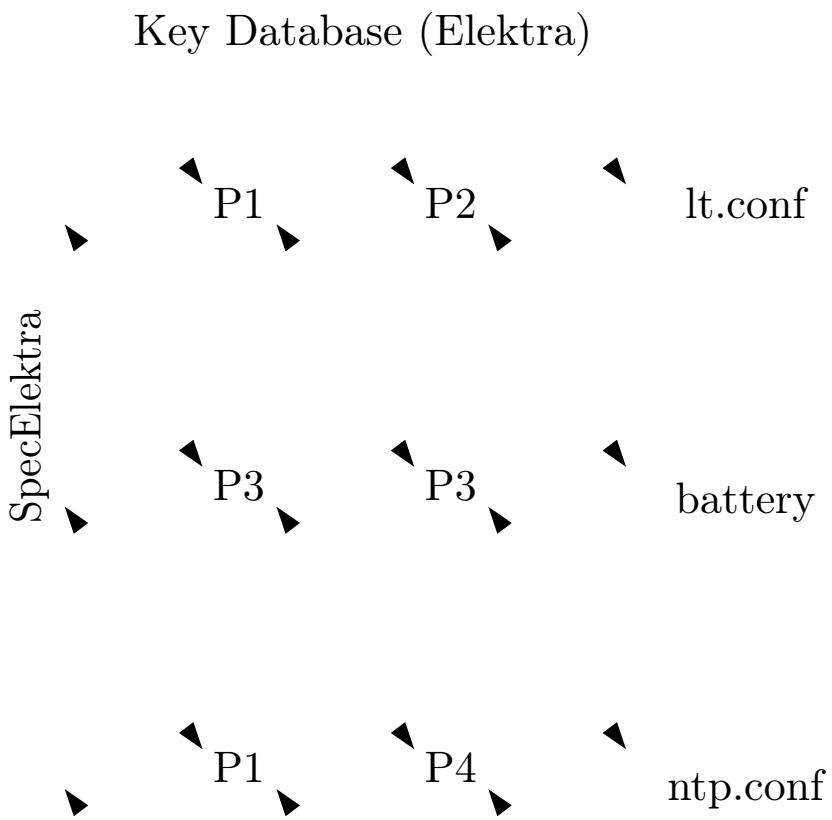
Elektra

Elektra as Virtual Filesystem

- configuration files are seen like “block devices”
- are mounted with respective filesystem drivers into the filesystem
- many tools and APIs evolved to work with files
- Idea of Elektra: establish a similar ecosystem for configuration

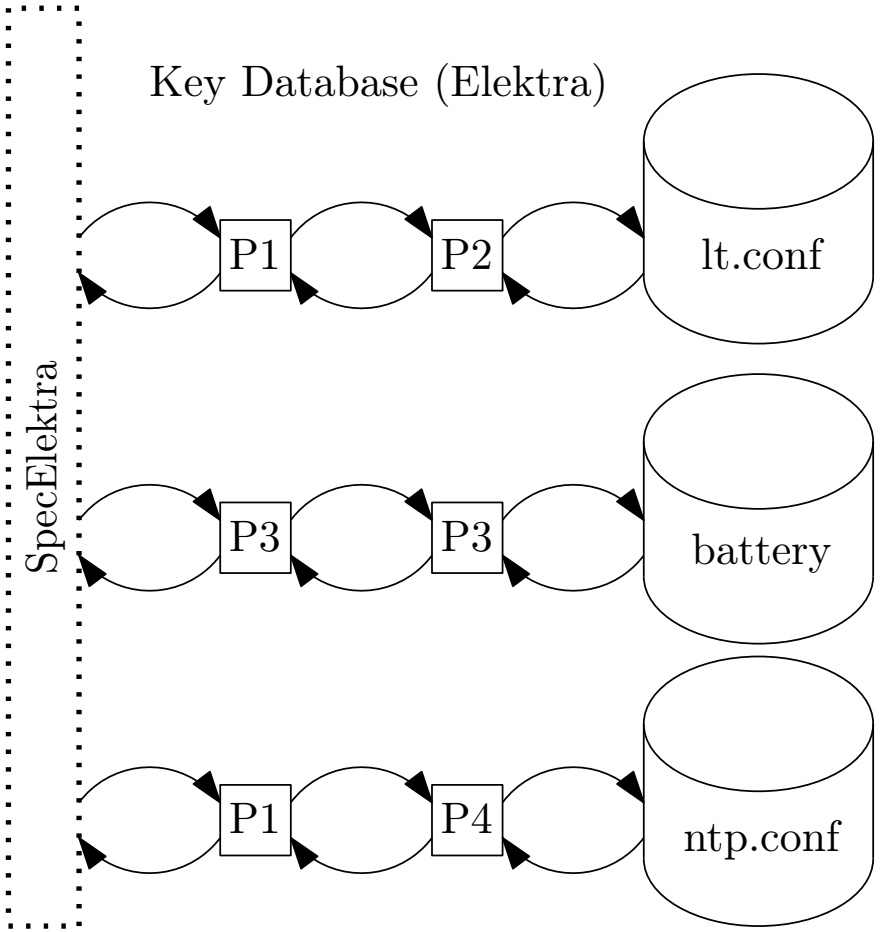
Why is Elektra not a Filesystem then?

- API semantics: key/value get/set
- namespaces: based on established semantics
- many features essential for misconfiguration hardening:
  - validation
  - visibility
  - defaults
  - ... (extensible specification)



Cylinders are configuration files, P? are plugins [17].
Key ideas:

- all work is done by plugins
- central data structure implements semantics

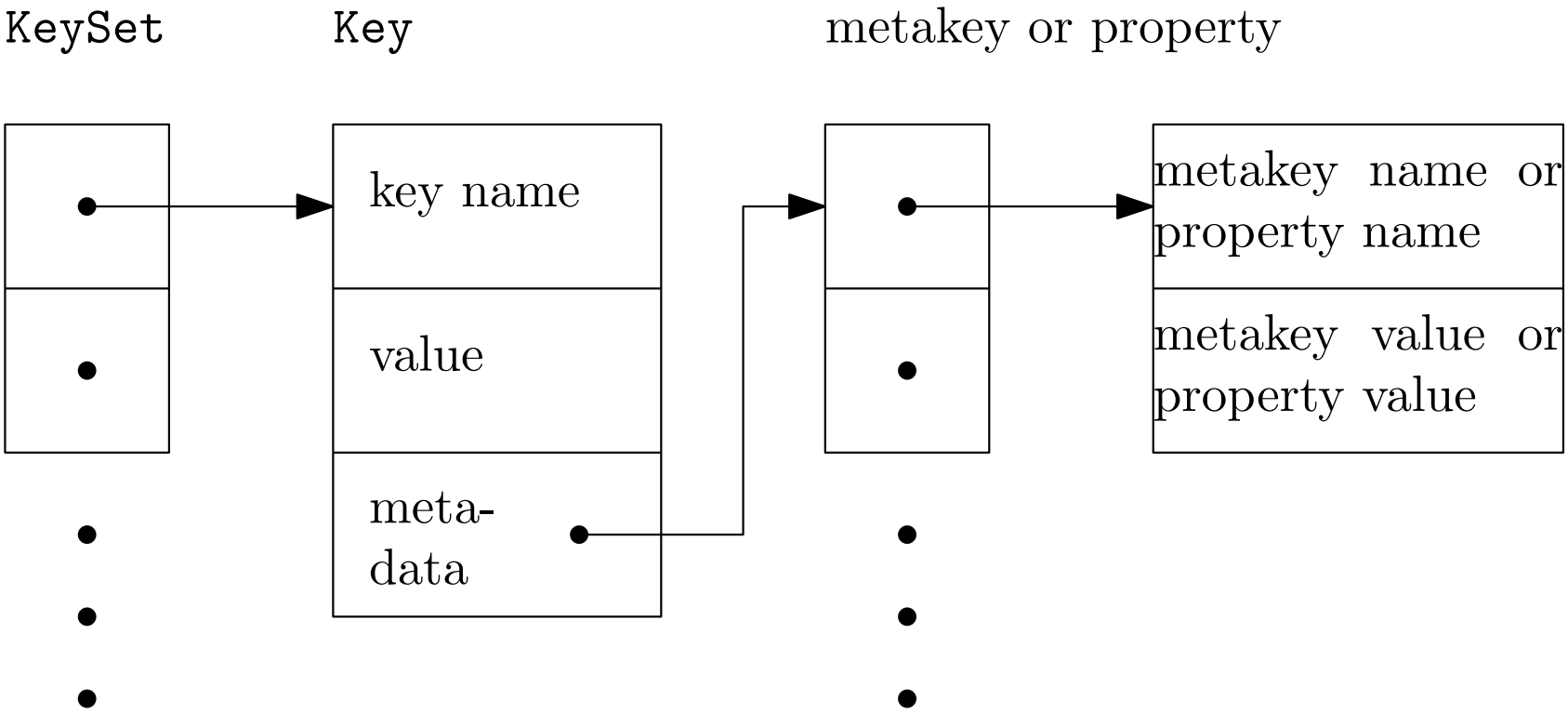


Cylinders are configuration files, P? are plugins [17].
Key ideas:

- all work is done by plugins
- central data structure implements semantics

KeySet

The common data structure between plugins:



Task

Is meta-data separated from or included in the data structure?

kdb.open(): The first step is to bootstrap into a situation where the necessary plugins can be loaded.

kdb.get(KeySet): The application (initially) fetches and (later) updates its configuration settings as a key set of type `KeySet` from the execution environment by one or many calls to `kdb.get`.

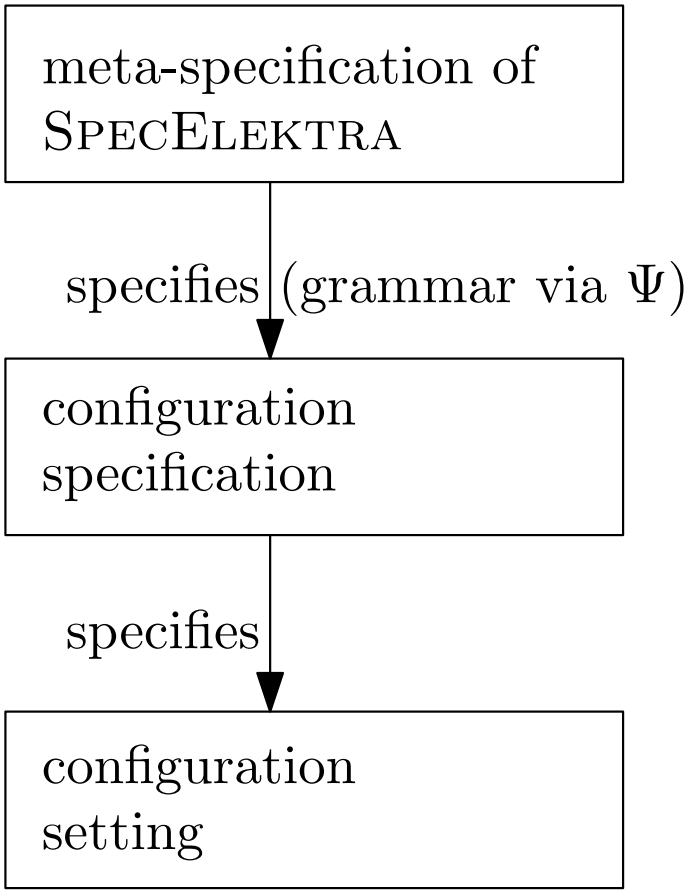
kdb.set(KeySet): When a user finishes editing configuration settings, `kdb.set` is in charge of writing all changes back to the key database.

kdb.close(): The last step is to close the connection to the key database.

Task

Break.

Metalevels



We will now walk through metalevels bottom-up.

Configuration Settings

A configuration file may look like:

```
1       a=5
2       b=10
3       c=15
```

We apply these configuration settings imperatively using:

```
1       kdb set /a 5
2       kdb set /b 10
3       kdb set /c 15
```

And we list them with `kdb ls /`.

Specifications

For specifications such as:

```
1       [slapd/threads/listener]
2         check/range :=1,2,4,8,16
3         default :=1
```

We apply the specifications imperatively using:

```
1       kdb setmeta /slapd/threads/listener\
2           check/range 1,2,4,8,16
3       kdb setmeta /slapd/threads/listener\
4               default 1
```

(automatically uses spec namespace)

Meta-Specifications

For meta-specifications such as:

```
[visibility]
type:=enum critical important user\
      advanced developer debug disabled
description:=Who should see this\
     configuration setting?
```

We apply the meta-specifications imperatively using:

```
kdb setmeta /elektra/meta/\
    visibility type enum ...
kdb setmeta /elektra/meta/\
    visibility description "Who ...
```

(see doc/METADATA.ini, disclaimer: 1.0 not yet released)

Task

Brainstorming: Ideas for (meta-)specifications.

Introspection

- unified get/set access to (meta*)-key/values
- access via applications, CLI, GUI, web-UI, ...
- GUI, web-UI can semantically interpret metadata
- access via any programming language
- access via any configuration management system

Users of Elektra

- Embedded systems
  - OpenWRT (distribution)
  - Broadcom (blue-ray devices)
  - Kapsch (cameras)
  - Toshiba (TVs)
- Server
  - Allianz (insurance)
  - TU Wien
  - puppet-libelektra
  - Other Universities
- Desktop
  - Oyranos
  - LCDproc (in progress)
  - KDE

Conclusion

- goals:
  - make simple configuration management tasks simple
  - improve robustness
  - improve extensibility (reusable plugins operating on key/value)
  - improve performance
  - good defaults
  - system-wide introspection
  - system-level dependency injection
- Elektra has no dependence to other libraries but only concrete plugins introduce dependences.

Abstractions

Abstraction

Requirement

A configuration library must be able to integrate (legacy) systems and must fully support (legacy) configuration files.

How can we deal with the many formats?

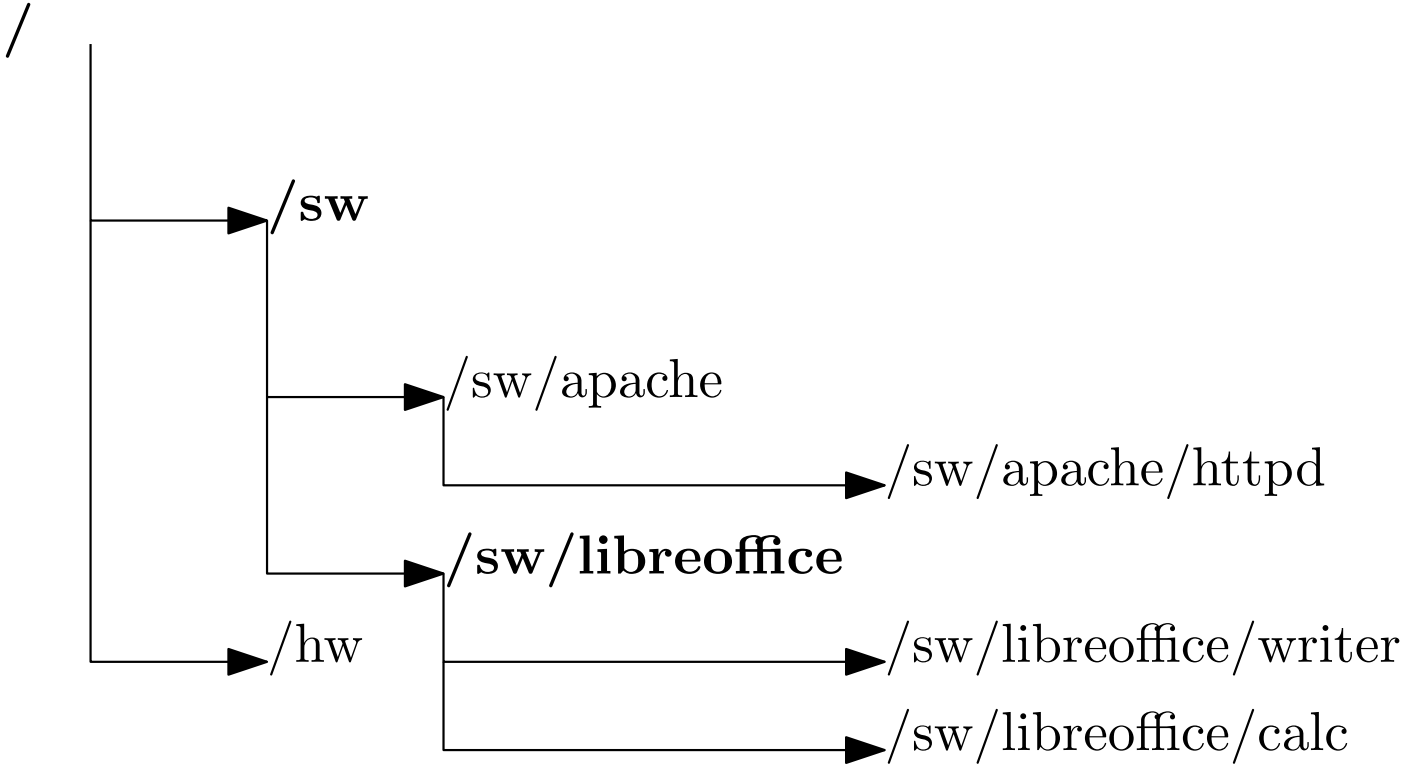
Key-Value

A key-value pair is the simplest generic data structure [23]. While all these formats above have many differences, all of them represent configuration settings as ***key-value pairs*** [11, 12, 21, 29].

For configuration as program you need to execute them first.

Mounting

Mounting integrates a backend into the key database [20]. Hence, Elektra allows several backends to deal with configuration files at the same time. Each backend is responsible for its own subtree of the key database.

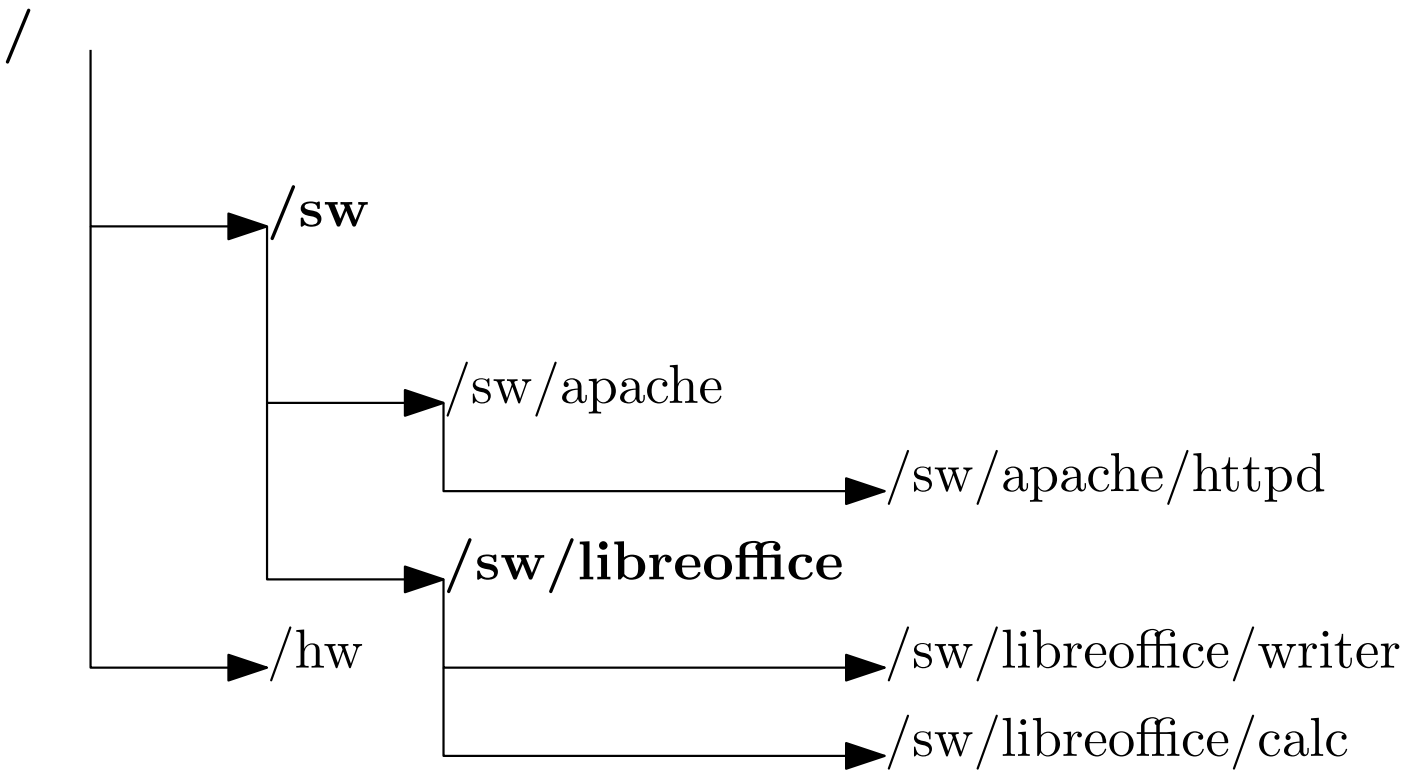


Plugins

Different backends can use different plugins:

`/sw` in the INI file config.ini

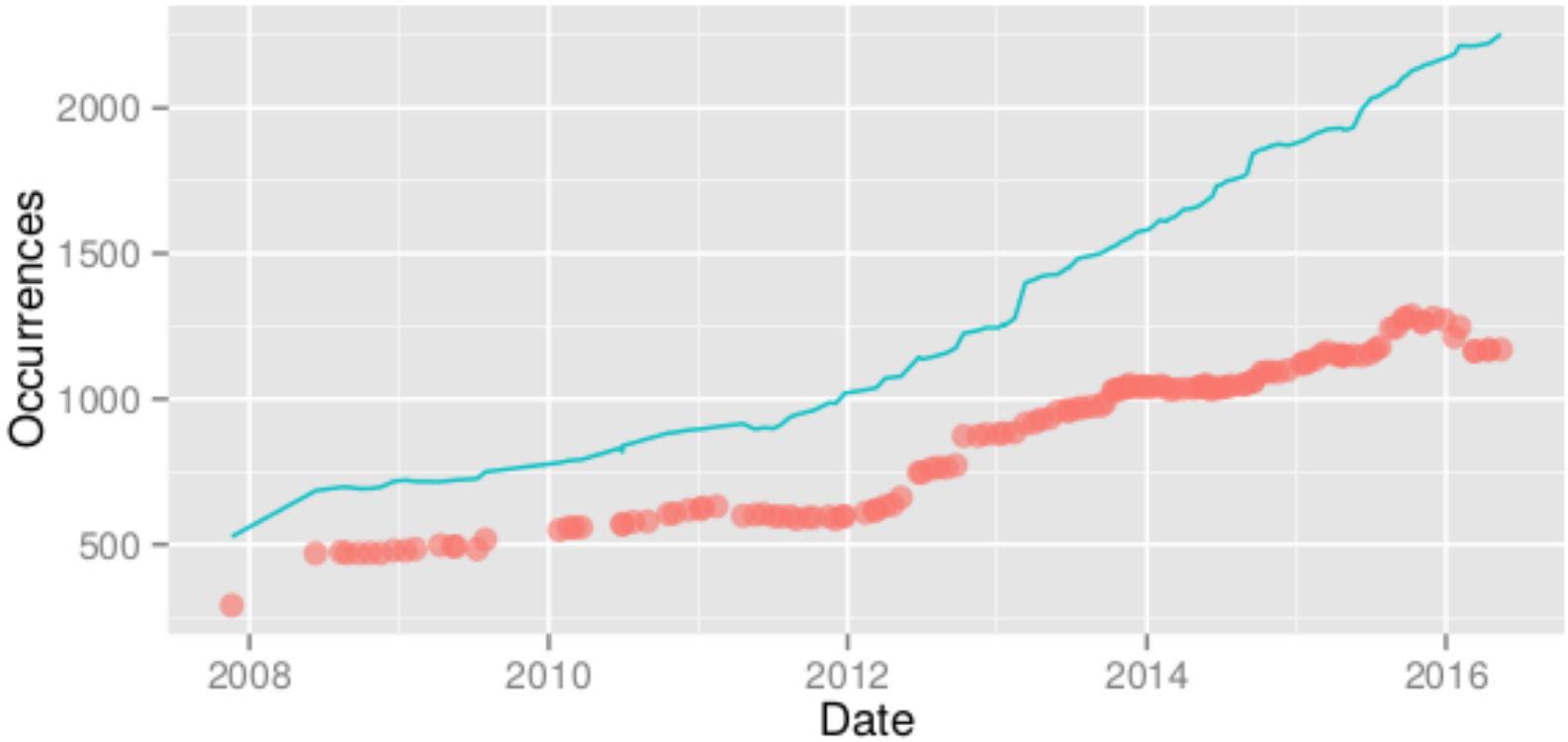
`/sw/libreoffice` in the XML file libreoffice.xml



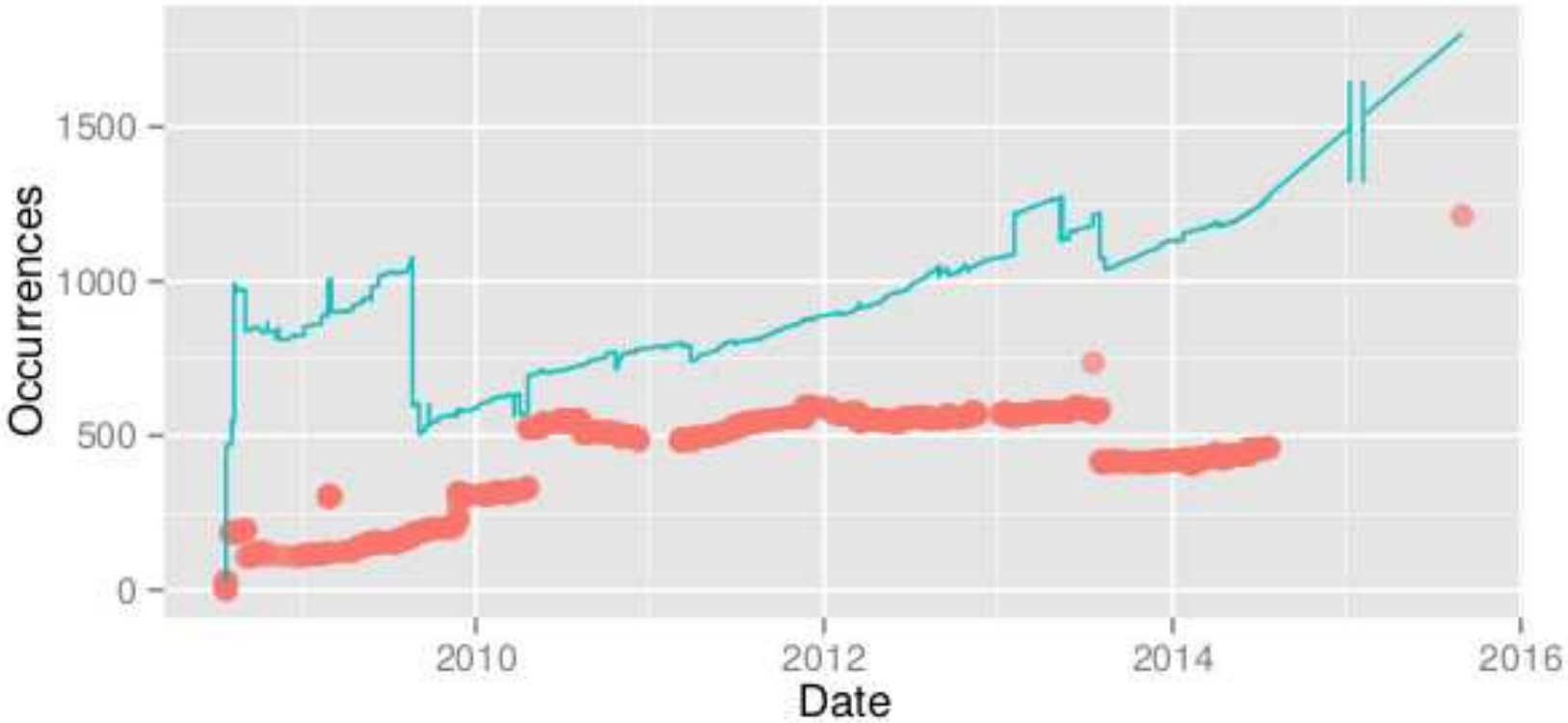
Task

Explain your neighbor what mounting is.

Trend Firefox



Trend Chromium



Outlook

- Environment Variables
- Command-line options
- Complexity

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

[2] Paul Anderson. Towards a high-level machine configuration system. In *LISA*, volume 94, pages 19–26, 1994.

[3] Richard Anthony, DeJiu Chen, Mariusz Pelc, Magnus Persson, and Martin Törngren. Context-aware adaptation in DySCAS. *Electronic Communications of the EASST*, 19, 2009. URL `http://gala.gre.ac.uk/5533/`.

[4] Lionel Cons and Piotr Poznanski. Pan: A high-level configuration language. In *LISA*, volume 2, pages 83–98, 2002. URL `http://static.usenix.org/events/lisa02/tech/full_papers/cons/cons_html/`.

[5] Fernando J. Corbató, Fernando H. Saltzer, and C. T. Clingen. Multics: The first seven years. In *Proceedings of the May 16-18, 1972, Spring Joint Computer Conference*, AFIPS '72 (Spring), pages 571–583, New York, NY, USA, 1972. ACM. doi: 10.1145/1478873.1478950. URL http://dx.doi.org/10.1145/1478873.1478950.

[6] Software Productivity Consortium Services Corporation. *Reuse-driven Software Processes Guidebook: SPC-92019-CMC, Version 02.00. 03*. Software Productivity Consortium Services Corporation, 1993.

[7] 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.

[8] Ian Goldberg, David Wagner, Randi Thomas, and Eric A. Brewer. A secure environment for untrusted helper applications: Confining the wily hacker. In *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography*, volume 6, pages 1–1, 1996.

[9] Sebastian Günther, Thomas Cleenewerck, and Viviane Jonckers. Software variability: the design space of configuration languages. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, pages 157–164. ACM, 2012. URL `http://dl.acm.org/citation.cfm?id=2110165`.

[10] Peng Huang, William J. Bolosky, Abhishek Singh, and Yuanyuan Zhou. ConfValley: a systematic configuration validation framework for cloud services. In *EuroSys*, page 19, 2015.

[11] 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.

[12] Neal Lathia, Kiran Rachuri, Cecilia Mascolo, and George Roussos. Open source smartphone libraries for computational social science. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 911–920, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2215-7. doi: 10.1145/2494091.2497345. URL http://dx.doi.org/10.1145/2494091.2497345.

[13] Zhenkai Liang, V. N. Venkatakrishnan, and R. Sekar. Isolated program execution: an application transparent approach for executing untrusted programs. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 182–191, December 2003. doi: 10.1109/CSAC.2003.1254323.

[14] Kim Mens, Rafael Capilla, Nicolas Cardozo, Bruno Dumas, et al. A taxonomy of context-aware software variability approaches. In *Workshop on Live Adaptation of Software Systems, collocated with Modularity 2016 conference*, 2016.

[15] Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki. Mining configuration constraints: static analyses and empirical results. In *ICSE*, pages 140–151, 2014.

[16] Jeff H. Perkins, Sunghun Kim, Sam Larsen, Saman Amarasinghe, Jonathan Bachrach, Michael Carbin, Carlos Pacheco, Frank Sherwood, Stelios Sidiroglou, Greg Sullivan, Weng-Fai Wong, Yoav Zibin, Michael D. Ernst, and Martin Rinard. Automatically patching errors in deployed software. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 87–102, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-752-3. doi: 10.1145/1629575.1629585. URL `http://dx.doi.org/10.1145/1629575.1629585`.

[17] Markus Raab. Improving system integration using a modular configuration specification language. In *Companion Proceedings of the 15th International Conference on Modularity*, MODULARITY Companion 2016, pages 152–157, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4033-5.

doi: 10.1145/2892664.2892691. URL http://dx.doi.org/10.1145/2892664.2892691.

[18] 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.

[19] 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.

[20] Markus Raab and Patrick Sabin. Implementation of Multiple Key Databases for Shared Configuration. `ftp://www.markus-raab.org/elektra.pdf`, March 2008. Accessed February 2014.

[21] Ariel Rabkin and Randy Katz. Static extraction of program configuration options. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 131–140. IEEE, 2011.

[22] Z. Cliffe Schreuders, Tanya Jane McGill, and Christian Payne. Towards usable application-oriented access controls: qualitative results from a usability study of SELinux, AppArmor and FBAC-LSM. *International Journal of Information Security and Privacy*, 6(1):57–76, 2012.

[23] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004*, September 2004. URL `http://elib.dlr.de/7444/`.

[24] Sander van der Burg and Eelco Dolstra. Automating system tests using declarative virtual machines. In *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, pages 181–190. IEEE, 2010.

[25] Jilles Van Gurp, Jan Bosch, and Mikael Svahnberg. On the notion of variability in software product lines. In *Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on*, pages 45–54. IEEE, 2001.

[26] Karina Villela, Adeline Silva, Tassio Vale, and Eduardo Santana de Almeida. A survey on software variability management approaches. In *Proceedings of the 18th International Software Product Line Conference - Volume 1*, SPLC '14, pages 147–156, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2740-4. doi: 10.1145/2648511.2648527. URL `http://dx.doi.org/10.1145/2648511.2648527`.

[27] Alexander von Rhein, Thomas Thüm, Ina Schaefer, Jörg Liebig, and Sven Apel. Variability encoding: From compile-time to load-time variability. *Journal of Logical and Algebraic Methods in Programming*, 85(1, Part 2):125–145, 2016. ISSN 2352-2208. doi: http://dx.doi.org/10.1016/j.jlamp.2015.06.007. URL `http://www.sciencedirect.com/science/article/pii/S2352220815000577`. Formal Methods for Software Product Line Engineering.

[28] Huai Wang and Wing Kwong Chan. Weaving context sensitivity into test suite construction. In *Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on*, pages 610–614, November 2009. doi: 10.1109/ASE.2009.79.

[29] Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Shankar Pasupathy. Do not blame users for misconfigurations. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 244–259. ACM, 2013.

[30] Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N. Bairavasundaram, and Shankar Pasupathy. An empirical study on configuration errors in commercial and open source systems. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 159–172, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0977-6. doi: 10.1145/2043556.2043572.

[31] Sai Zhang and Michael D. Ernst. Automated diagnosis of software configuration errors. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 312–321, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4673-3076-3.

[32] Sai Zhang and Michael D. Ernst. Which configuration option should I change? In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 152–163, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2756-5. doi: 10.1145/2568225.2568251. URL http://dx.doi.org/10.1145/2568225.2568251.