

# Configuration Management

Markus Raab

Institute of Information Systems Engineering, TU Wien

29.5.2018



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: PR for first issue done, second started

03.04.2019: first issue done, PR for second

10.04.2019: mid-term submission of exercises

08.05.2019: different location: Complang Library

15.05.2019:

22.05.2019: all 5 issues done

29.05.2019:

05.06.2019: final submission of exercises

12.06.2019:

19.06.2019: last corrections of exercises

26.06.2019: exam

# Tasks for today

(until 29.05.2019 23:59)

## Task

Continue teamwork and homework. Make a clear description for what your team partner should do.

## Tasks for next week

(until 05.06.2019 23:59)

### Task

Submit teamwork and homework.

## 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

# Learning Outcomes

Students will be able to

- remember basics of context-awareness.
- remember basic characteristics of key databases.
- remember the history of configuration management.

# Introspection (Recapitulation)

## Task

What is internal and external specification? What is introspection?

- *internal*: within applications' source code
- *introspection*: unified get/set access to (meta\*)-key/values
- access via applications, CLI, GUI, web-UI, ...
- access via any programming language (similar to file systems)
- GUI, web-UI can semantically interpret metadata
- assemble modular parts (validation, logging, ...)
- needed as communication between producers and consumers
- essential for *no-futz computing* Holland et al. [19]

# Example Documentation (Recapitulation)

```
1 [slapd/threads/listener]
2   check/range:=1,2,4,8,16
3   default:=1
4   description:=adjust to use more threads
5   rationale:=needed for many-core systems
6   requirement:=1234
7   visibility:=user
```



## Reevaluate specifications (Recapitulation)

### Task

In which situations should you reevaluate if a configuration setting (specification) is needed?

- 1 a requirement,
- 2 an architectural decision,
- 3 a technical need, and
- 4 an ad hoc decision.

### Goal

Reduction of all not-needed configuration settings (user view).

# Semantic three-way merge (Recapitulation)

## 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-Awareness

- 1 Context-Awareness
- 2 Key Databases
- 3 History of Configuration Management

Khalil and Connelly [25] conducted a study where all users found context-aware configuration (very) useful. They learned that in 89 % of cases the mapping between activities and settings was consistent for individual users. In the study, context-aware configuration improved satisfaction, even if deduced settings sometimes were not appropriate. For example, a participant stated:

*"I like how it changes state without you having to tell it to. I always forget to turn my cell [off] in class and turn it on after."*

## Definition (Recapitulation)

As adapted from Chalmers [10]:

***Context** is the circumstances relevant to the configuration settings of the application.*

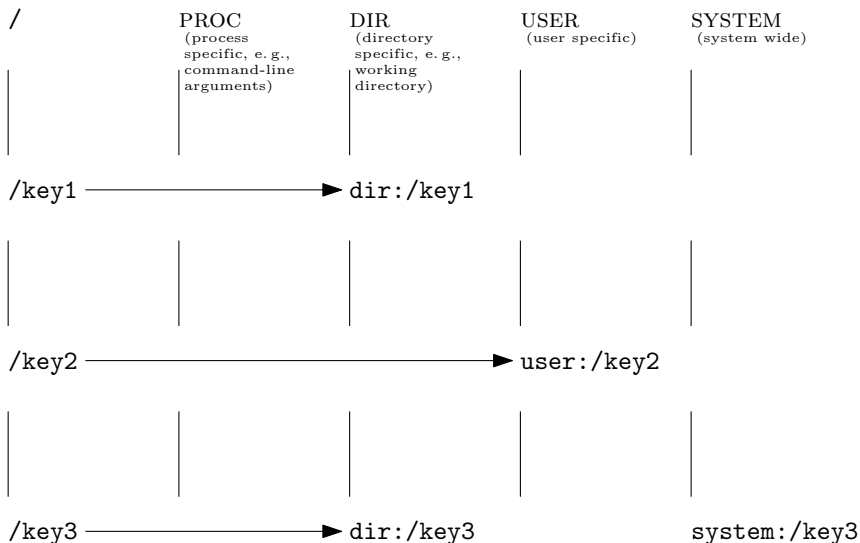
We extend the definition with:

***Context-aware configurations** are configuration settings that are consistent with its context. **Context-aware configuration access** is configuration access providing context-aware configuration.*

# Types of Configuration (Recapitulation)

- Valid configuration** does not contradict the present validation specifications. With a valid configuration, applications can start but they may not do what the user wanted or may be inconsistent with context.
- Suitable configuration** is valid with respect to additional specifications from the user that describe the system the user requires [26].
- Optimal configuration** is optimal with respect to given optimization criteria. Optimization criteria are important if managing configuration of many computers but are rarely needed for configuration access discussed in this book.
- Context-aware configuration** is in accordance with its context. Unlike configuration settings, the context changes in ways outside of our control.

# Cascading (Recapitulation)



# Context-oriented Programming

One of the many systematic ways to write context-aware applications is called ***context-oriented programming*** [1, 5–8, 11, 12, 14, 18, 22–24, 28, 32–37]. Contrary to other techniques to improve context awareness, it focuses on the language level. Its run-time system is rather small, it does not need sophisticated frameworks, databases, or middleware. Context-oriented programming supports implementation of context-aware applications.



# Contextual Values

Tanter [35] introduced a lightweight extension to context-oriented programming: **Contextual values** are variables whose values depend on the context in which they are read and modified. They *“boil down to a trivial generalization of the idea of thread-local values”*. The key idea is to use layers as *“discriminate amongst possible values, not only the current thread”* [35]. Side effects are limited to the respective context [29].

## Contextual Values (Pseudocode)

```
1 void printBrowserConfig (Config config)
2 {
3     context.with("private")
4     {
5         println (config.keepHistory);
6     }
7     // same thread, different context:
8     println (config.keepHistory);
9
10    context.activate(currentLocation)
11 }
```

# Introspection vs. Code Generation (Partly Recapitulation)

Implementation of contextual values might be in key database or in generated code. Advantages/Disadvantages?

- more techniques for performance improvements with code generation
- + specification can be updated live on the system without recompilation
- + tooling has generic access to all specifications
- + new features the key database (e.g., better validation) are immediately available consistently
- needed if context differs within same thread

## Implication

We generally prefer introspection, except for a very thin configuration access API.

Task

Break.

## Context Specifications (Recapitulation)

- Determine threads from CPUs:

```
1 [env/layer/cpu]
2   type := long
3 [slapd/threads/listener]
4   context := /slapd/threads/%cpu%/listener
```

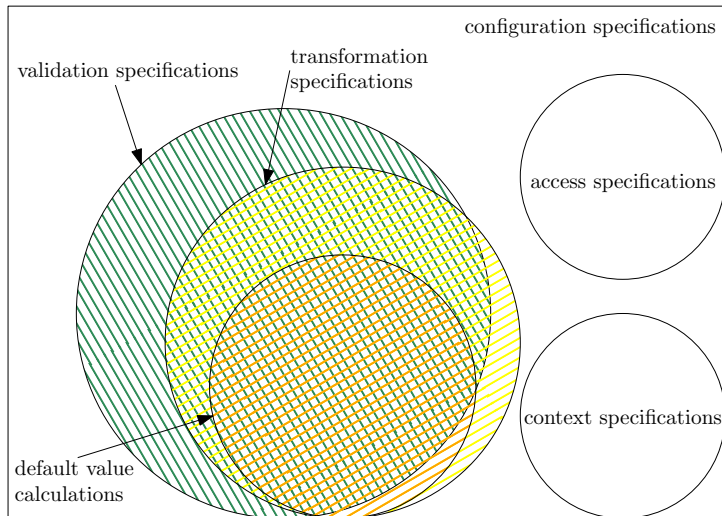
- Determine vibration from sensors:

```
1 [phone/call/vibration]
2   type := boolean
3   context := /phone/call/%inocket%/vibration
```

- Determine proxy settings from network:

```
1 [env/override/http_proxy]
2   context := /http_proxy/%interface%/%network%
```

# Types of Specifications (Recapitulation)



## Key Databases

- 1 Context-Awareness
- 2 Key Databases
- 3 History of Configuration Management

# Status Quo

## Applications . . .

- usually consume configuration settings from configuration files, command-line arguments, and environment variables.
- sometimes have a single GUI or CLI for configuration settings.
- rarely have an API to access configuration settings.
- rarely consider context.
- rarely do in-depth validation.
- nearly never have an API to access configuration specifications.

### Task

Think about applications you know. Discuss it with your neighbor.



# Examples

- Postfix<sup>1</sup>: CLI (Properties, CSV, and others)
- KDE<sup>2</sup>: GUI, CLI (INI)
- Libreoffice: GUI (XML)
- Firefox [21]: GUI (JavaScript and others)
- sudo: CLI (sudoers edited with visudo)
- X.org<sup>3</sup>: xorg.conf
- gpsd<sup>4</sup>: environment variables and command-line arguments

---

<sup>1</sup><http://www.postfix.org/OVERVIEW.html>

<sup>2</sup><https://api.kde.org/frameworks/kconfig/html/>

<sup>3</sup><ftp://www.x.org/pub/X11R6.7.0/doc/xorg.conf.5.html>

<sup>4</sup><http://www.aosabook.org/en/gpsd.html>

# Design Decisions

There are many ways to design configuration access but many decisions are only pragmatic and irrelevant with proper key/value abstraction.

## Task

Which design decisions are there? Why are they (ir)relevant?

- Which configuration file format? (irrelevant due to key/values)
- Split up into multiple configuration files? (irrelevant due to 3-way merging)
- Where are the configuration files? (irrelevant due to mounting and resolver)
- Important: Introspection, Validation, Horizontal Modularity, Integration, Specification, API, Guarantees, ...

## Key Databases (Usage)

*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 %,  $n = 218$ )
- configuration files (74 %,  $n = 218$ )
- Freedesktop standards (20 %,  $n = 205$ )
- Windows Registry (13 %) ( $\leq 13$  %,  $n \geq 185$ ) [talk later]
- X/Q/GSettings (4 %, 11 %, 9 %)
- KConfig (5 %)
- dconf (7 %)
- plist (7 %)

# Distributed Key Databases

Examples:

- Redis: in-memory with persistence and notification
- Zookeeper
- etcd:
  - not in-memory
  - get/set/watch interface via REST
  - distributed coordination [27]
  - needs configuration itself

# Elektra

- is not only a key database but a specification language to describe a key database
- plugins implement the specification (could be distributed but focus is configuration files)
- is library based (no single point of failure, no distributed coordination needed)
- supports transactions (persisting whole KeySets at once)
- supports integration of existing configuration

# History of Configuration Management

- 1 Context-Awareness
- 2 Key Databases
- 3 History of Configuration Management

# Definition

## *Configuration Management:*

- is a discipline in which configuration (in the broader sense) is administered.
- makes sure computers are assembled from desired parts and the correct applications are installed.
- ensures that the execution environment of installed applications is as required.

# Definition

## *Configuration management tools:*

- help people involved in configuration management.
- have means to describe the desired configuration of the whole managed system.
- try to converge the actual configuration to the desired one [9].



## Challenging tasks in configuration management:

- inventory list
- installing packages
- monitoring
- add/replace machines
- maintaining files/databases/...
- *configuration file manipulation*

# Cloning

It all started with:

- clone all files with dd, rdist, rsync or unison (“golden image”)
- then do necessary modifications with scripts or profiles
  - + works very good for many identical stateless machines
  - fails if differences between machines are too big

# Scripts

First improvement: have a script to create the “golden image”.  
Possible benefits:

- Documentation
- Customization (using configuration settings)
- **Reproducibility**: Reproduce creation using different operating system versions

# Profiles

**Profiles** are groups of configuration settings between which the user can easily switch.

- by hostname, information EEPROM, manual selection, ...
- can be activated as context:

```
1 [%application%/profile]
2   type := string
3   opt := p
4   opt/long := profile
5   default :=
```

# First four configuration management tools

Cloning, and then NIS/NFS, was state of the art for a long time, until in 1994 when *“the community nearly exploded with four new configuration systems”* [13]:

- lcfg** from Anderson [3]. The development of lcfg started first in 1991 [2, 3]. Nevertheless, its development still continues [4, 17].

- GeNUAdmin** from Harlander [15].

- omniconf** from Hideyo [16].

- config** from Rouillard and Martin [31].

## Possible Benefits

- All advantages scripts have:  
Documentation, Customization, Reproducibility
- Declarative description of the system  
(Infrastructure as Code [20])
- Less configuration drift
- Error handling
- Pull/Push
- Reusability
- (Resource) Abstractions

# Conclusion

- Context-awareness is a goal.
- Contextual values is a way to implement it.
- Many (distributed) key databases enable us to persist configuration settings.
- Definition and challenges in configuration management.
- Cloning: There and back again.

# Outlook

- Challenges in Configuration Management
- Properties: self-describing, idempotent, round-tripping
- Validation
- Configuration management languages



- [1] Unai Alegre, Juan Carlos Augusto, and Tony Clark. Engineering context-aware systems and applications: A survey. *Journal of Systems and Software*, 117:55–83, 2016. ISSN 0164-1212. doi: 10.1016/j.jss.2016.02.010. URL <http://www.sciencedirect.com/science/article/pii/S0164121216000467>.
- [2] Paul Anderson. Local system configuration for syssies. Technical report, CS-TN-38, Department of Computer Science, University of Edinburgh, Edinburgh, 1991.
- [3] Paul Anderson. Towards a high-level machine configuration system. In *LISA*, volume 94, pages 19–26, 1994.
- [4] Paul Anderson, Alastair Scobie, et al. Lcfig: The next generation. In *UKUUG Winter conference*, pages 4–7, 2002.

- [5] Tomoyuki Aotani, Tetsuo Kamina, and Hidehiko Masuhara. Unifying multiple layer activation mechanisms using one event sequence. In *Proceedings of 6th International Workshop on Context-Oriented Programming*, COP'14, pages 2:1–2:6, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2861-6. doi: 10.1145/2637066.2637068. URL <http://dx.doi.org/10.1145/2637066.2637068>.
- [6] Malte Appeltauer, Robert Hirschfeld, and Tobias Rho. Dedicated programming support for context-aware ubiquitous applications. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, Second UBICOMM.*, pages 38–43. IEEE, 2008.

- [7] Malte Appeltauer, Robert Hirschfeld, Michael Haupt, Jens Lincke, and Michael Perscheid. A comparison of context-oriented programming languages. In *International Workshop on Context-Oriented Programming*, COP '09, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-538-3. doi: 10.1145/1562112.1562118. URL <http://dx.doi.org/10.1145/1562112.1562118>.
- [8] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.
- [9] Mark Burgess. A site configuration engine. In *USENIX Computing systems*, volume 8, pages 309–337, 1995.
- [10] Daniel Chalmers. *Contextual mediation to support ubiquitous computing*. PhD thesis, University of London, 2002.

- [11] Pascal Costanza, Robert Hirschfeld, and Wolfgang De Meuter. Efficient layer activation for switching context-dependent behavior. In David E. Lightfoot and Clemens Szyperski, editors, *Modular Programming Languages*, volume 4228 of *Lecture Notes in Computer Science*, pages 84–103. Springer, 2006. ISBN 978-3-540-40927-4. URL [http://dx.doi.org/10.1007/11860990\\_7](http://dx.doi.org/10.1007/11860990_7).
- [12] Anind K. Dey and Gregory D. Abowd. The what, who, where, when, why and how of context-awareness. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, NY, 2000. ACM. ISBN 1-58113-248-4. URL <ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf>.
- [13] Rémy Evard. An analysis of UNIX system configuration. In *LISA*, volume 97, pages 179–194, 1997.

- [14] Sebastián González, Nicolás Cardozo, Kim Mens, Alfredo Cádiz, Jean-Christophe Libbrecht, and Julien Goffaux. *Subjective-C*, pages 246–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-19440-5. doi: 10.1007/978-3-642-19440-5\_15. URL [http://dx.doi.org/10.1007/978-3-642-19440-5\\_15](http://dx.doi.org/10.1007/978-3-642-19440-5_15).
- [15] Magnus Harlander. Central system administration in a heterogeneous Unix environment: GeNUAdmin. In *LISA VIII Proceedings*, 1994.
- [16] Imazu Hideyo. OMNICONF—making os upgrades and disk crash recovery easier. In *LISA VIII Proceedings*, 1994.
- [17] Johannes Hintsch, Carsten Görling, and Klaus Turowski. A review of the literature on configuration management tools. 2016.

- [18] Robert Hirschfeld, Hidehiko Masuhara, Atsushi Igarashi, and Tim Felgentreff. Visibility of context-oriented behavior and state in I. In *Proceedings of the 31th JSSST Annual Conference*, pages 2–1, 2014.
- [19] David A. Holland, William Josephson, Kostas Magoutis, Margo I. Seltzer, Christopher A. Stein, and Ada Lim. Research issues in no-futz computing. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 106–110. IEEE, May 2001. doi: 10.1109/HOTOS.2001.990069.
- [20] Waldemar Hummer, Florian Rosenberg, Fábio Oliveira, and Tamar Eilam. Testing idempotence for infrastructure as code. In David Eysers and Karsten Schwan, editors, *Middleware 2013*, pages 368–388, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-45065-5.

- [21] 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>.
- [22] Hong Jong-yi, Suh Eui-ho, and Kim Sung-Jin. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522, 2009. ISSN 0957-4174. URL <http://dx.doi.org/10.1016/j.eswa.2008.10.071>.

- [23] Tetsuo Kamina, Tomoyuki Aotani, Hidehiko Masuhara, and Tetsuo Tamai. Context-oriented software engineering: A modularity vision. In *Proceedings of the 13th International Conference on Modularity*, MODULARITY '14, pages 85–98, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2772-5.
- [24] Roger Keays and Andry Rakotonirainy. Context-oriented programming. In *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDe '03, pages 9–16, New York, NY, USA, 2003. ACM. ISBN 1-58113-767-2. doi: 10.1145/940923.940926. URL <http://dx.doi.org/10.1145/940923.940926>.
- [25] Ashraf Khalil and Kay Connelly. *Context-Aware Configuration: A Study on Improving Cell Phone Awareness*, pages 197–209. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31890-3. doi: 10.1007/11508373\_15. URL [http://dx.doi.org/10.1007/11508373\\_15](http://dx.doi.org/10.1007/11508373_15).



- [26] Manuele Kirsch-Pinheiro, Raúl Mazo, Carine Souveyet, and Danillo Sprovieri. Requirements analysis for context-oriented systems. *Procedia Computer Science*, 83:253–261, 2016. ISSN 1877-0509. doi: <http://dx.doi.org/10.1016/j.procs.2016.04.123>. URL <http://www.sciencedirect.com/science/article/pii/S1877050916301466>. The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops.
- [27] Diego Ongaro and John K Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–319, 2014.

- [28] John Plaice and Blanca Mancilla. The cartesian approach to context. In *Proceedings of the 2nd International Workshop on Context-Oriented Programming*, COP '10, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0531-0. doi: 10.1145/1930021.1930024. URL <http://dx.doi.org/10.1145/1930021.1930024>.
- [29] Markus Raab. Unanticipated context awareness for software configuration access using the getenv API. In *Computer and Information Science*, pages 41–57. Springer International Publishing, Cham, 2016. ISBN 978-3-319-40171-3. doi: 10.1007/978-3-319-40171-3\_4. URL [http://dx.doi.org/10.1007/978-3-319-40171-3\\_4](http://dx.doi.org/10.1007/978-3-319-40171-3_4).

- [30] 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.
- [31] John P. Rouillard and Richard B. Martin. Config: A mechanism for installing and tracking system configurations. In *LISA*, 1994.
- [32] Guido Salvaneschi, Carlo Ghezzi, and Matteo Pradella. Context-oriented programming: A software engineering perspective. *Journal of Systems and Software*, 85(8): 1801–1817, 2012. ISSN 0164-1212. URL <http://dx.doi.org/10.1016/j.jss.2012.03.024>.

- [33] Hans Schippers, Tim Molderez, and Dirk Janssens. A graph-based operational semantics for context-oriented programming. In *Proceedings of the 2nd International Workshop on Context-Oriented Programming, COP '10*, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0531-0. doi: 10.1145/1930021.1930027. URL <http://dx.doi.org/10.1145/1930021.1930027>.
- [34] Matthias Springer, Hidehiko Masuhara, and Robert Hirschfeld. Classes as layers: Rewriting design patterns with COP: Alternative implementations of decorator, observer, and visitor. In *Proceedings of the 8th International Workshop on Context-Oriented Programming, COP'16*, pages 21–26, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4440-1. doi: 10.1145/2951965.2951968. URL <http://dx.doi.org/10.1145/2951965.2951968>.

- [35] Éric Tanter. Contextual values. In *Proceedings of the 2008 Symposium on Dynamic Languages*, DLS '08, pages 3:1–3:10, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-270-2. doi: 10.1145/1408681.1408684. URL <http://dx.doi.org/10.1145/1408681.1408684>.
- [36] Martin von Löwis, Marcus Denker, and Oscar Nierstrasz. Context-oriented programming: Beyond layers. In *Proceedings of the 2007 International Conference on Dynamic Languages*, ICDL '07, pages 143–156, New York, NY, USA, 2007. ACM. ISBN 978-1-60558-084-5. URL <http://dx.doi.org/10.1145/1352678.1352688>.

- [37] Benjamin Hosain Wasty, Amir Semmo, Malte Appeltauer, Bastian Steinert, and Robert Hirschfeld. ContextLua: Dynamic behavioral variations in computer games. In *Proceedings of the 2nd International Workshop on Context-Oriented Programming, COP '10*, pages 5:1–5:6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0531-0. doi: 10.1145/1930021.1930026. URL <http://dx.doi.org/10.1145/1930021.1930026>.