

Unanticipated Context Awareness for Software Configuration Access using the getenv API

Markus Raab

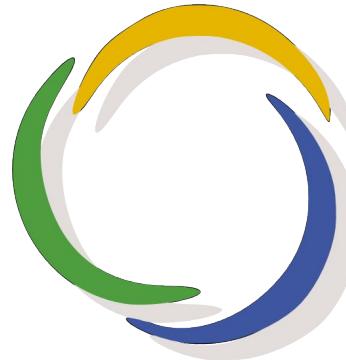
Vienna University of Technology

Institute of Computer Languages, Austria

Email: markus.raab@complang.tuwien.ac.at

Outline

- **Motivation**
 - Background
 - Goals
 - Research Questions
- **EnvElektra**
- **Evaluation**
 - Contributions
 - Benchmarks
- Conclusion



Elektra's Logo

Example

- mobile device
- many contexts
 - location
 - inmeeting
 - outdoor
 - inpocket
- desired behavior depends on context
- context often not **considered**
 - (although it is **available** by sensors)



Context-Oriented Programming: Layers

- originates from object-oriented programming
- layers represents context
- can be activated anywhere in the program
 - dynamic scope



many layers
can be active

```
void rcvPhoneCall() {  
    e.context().with()<Location>() ([&] {  
        vibrate();  
    });  
    // vibrate();  
}
```

name of layer

part of dynamic scope

Contextual Values

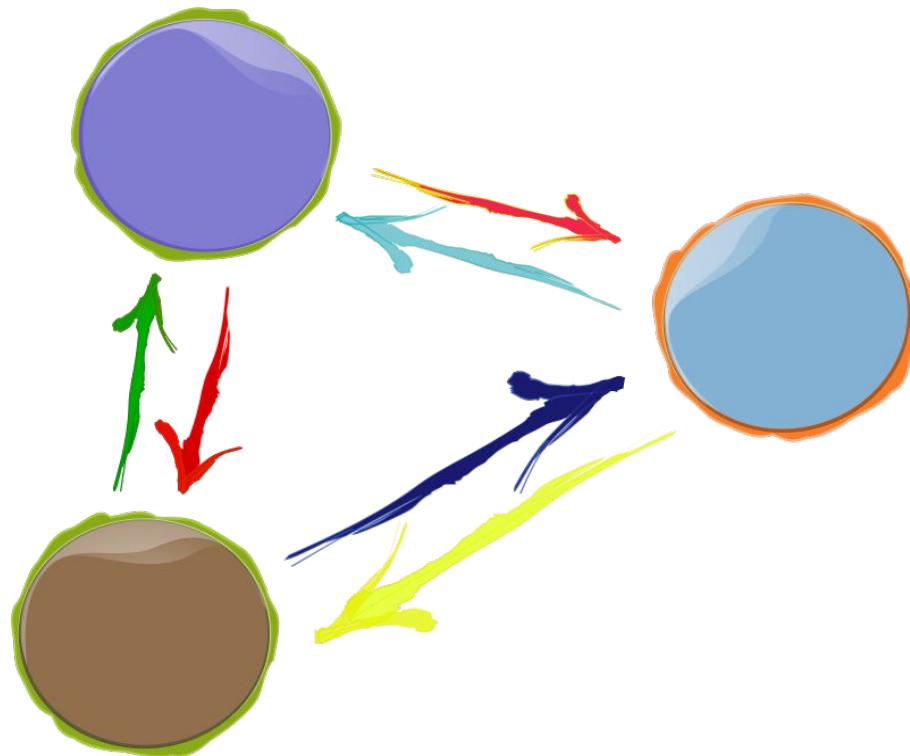
- “trivial generalization of thread-local values” with layers
- use dynamic scoping as in context-oriented programming
- use and access performance identical to variables

```
void visit (Person & p) {  
    p.context().with()<Location>() ([&] {  
        cout << "visit " << ++p.visits  
        << "in" << p.greeting;  Griaß enk  
    });  
    cout << p.greeting;  今日は  
}
```

Different Context

Goals

- **Context-Aware**
e.g. location
- **Customizable**
adapt to user
- **Reuse**
use unmodified applications



Research Questions

- **RQ1:** What are the usage patterns of `getenv()` in popular applications?
- **RQ2:** For which applications can we actually exploit `getenv()` to be used for unanticipated context awareness? What are the fundamental limitations?
- **RQ3:** What is the overhead that occurs in a system using EnvElektra?

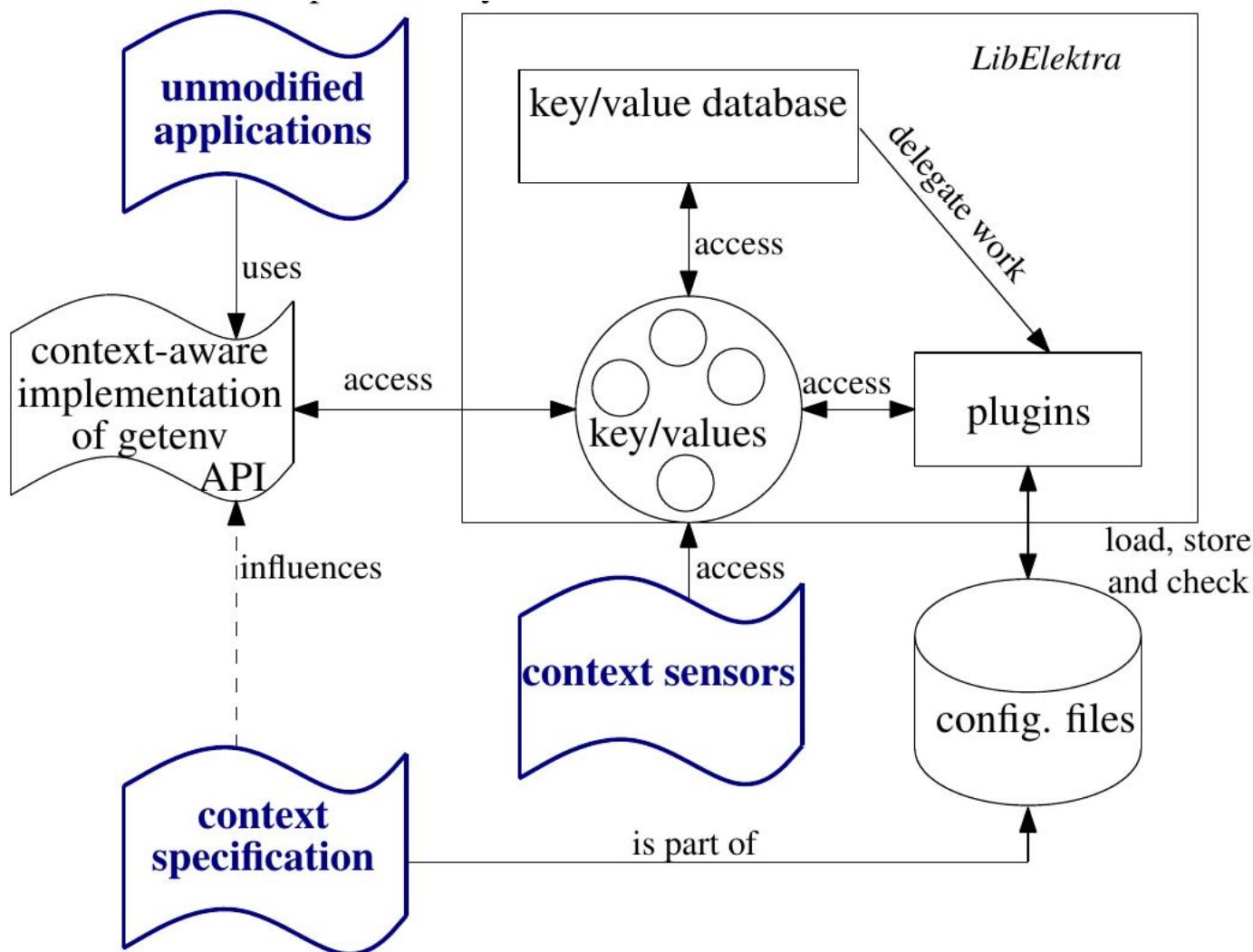


TECHNISCHE
UNIVERSITÄT
WIEN

Vienna University of Technology

EnvElektra

Architecture



Context Specification

- within configuration
- getenv("vibration")
- Context Specification:

```
[/env/vibration]
```

type=boolean

- /: denotes hierarchy of contextual values
- needed for **customization**
 - initialize and persist every contextual value



Contextual Lookup

- Context Specification:

```
[/env/vibration]
```

type=boolean

context=/phone/call/%location%/vibration



- %: placeholders for layers
- adaptable configuration

```
/phone/call/inmeeting/vibration = off  
/phone/call/outdoor/vibration = off  
/phone/call/inpocket/vibration = on
```



Evaluation

Case Studies

- applied EnvElektra on 15 applications/systems
- works very well if:
 - getenv() is called often (also indirectly)
 - external applications are executed
- needs restart otherwise
- examples:
 - proxy for browsers/wget/curl
 - gimp open/help
 - UNIX man utiliy



Usage Patterns

application	lines of code	getenv all	getenv init	all unique	later unique	same
akonadi	37,214	10,357	8655	110	12	5126
chromium	18,032,183	6006	1803	1118	192	165
curl	249,380	19	8	12	8	4
eclipse	3,311,712	2790	2696	389	42	1495
evolution	672,789	4407	1488	1060	24	163
firefox	12,394,938	3371	2049	276	70	895
gimp	901,703	2551	1115	217	137	364
inkscape	479,849	722	457	160	51	166
libreoffice	5,482,215	3354	2891	258	59	1493
lynx	192,012	1931	961	27	27	923
man	142,183	2862	13	86	76	2
smplayer	76,170	212	164	71	8	53
wget	142,603	11	10	8	1	3
Mean	3,217,074	2969	1716	292	54	835
Median	479,849	2790	1115	160	42	166
Total	41,821,956	38,593	22,310	3792	707	10,852
KDE	*	*	9606	265	*	2634
GNOME	*	*	144	47	*	4
Debian	*	*	5317	430	*	286

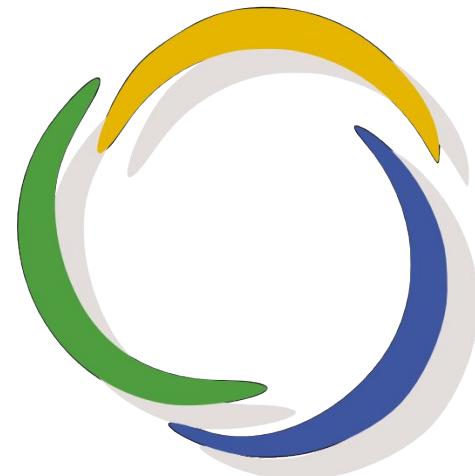
Overhead

- without run-time changes 2.6% (gimp)
to 14% (compilation)
- curl http-proxy-transition
 - dynamic reload 10%
 - per-change 4%



Source Code

- source code released as free software within Elektra
 - >60 predefined plugins
 - support for hundreds kinds of configuration files
 - specification is also configuration (e.g. in XML, JSON)
- <http://www.libelektra.org>
 - version 0.8.17 released at 14. June 2016



Conclusion

- context-awareness without modifications
- getenv used pervasive
- external context specification
- evaluation
 - can be exploited in **many** cases
 - acceptable overhead

Markus Raab

Vienna University of Technology

Institute of Computer Languages, Austria

Email: markus.raab@complang.tuwien.ac.at

<http://elektra.limequery.org/625192>

Thank you for your
attention!

Markus Raab

Vienna University of Technology

Institute of Computer Languages, Austria

Email: markus.raab@complang.tuwien.ac.at

Benchmark Setup

- Laptop: hp ® EliteBook 8570w ™
 - CPU Intel ® Core i7-3740QM @ 2.70GHz
 - 7939 MB Ram
- GNU/Linux Debian Wheezy 7.5
- gcc compiler Debian 4.7.2-5
 - with the options `-std=c++11, -O2`
- measured the time using `gettimeofday`
- Median of eleven executions

Related Work

context variables (check on every usage)

M. von Löwis, M. Denker, and O. Nierstrasz, “Context-oriented programming: Beyond layers,” in Proceedings of the 2007 International Conference on Dynamic Languages

ensure-active-layers (global layer activation)

P. Costanza, R. Hirschfeld, and W. De Meuter, “Efficient layer activation for switching context-dependent behavior,” in Modular Programming Languages

partial evaluation avoids usage of libxml2

M. Jung, R. Laue, and S. A. Huss, “A case study on partial evaluation in embedded software design,” in SEUS 2005

hybrid mediator-observer pattern

O. Riva, C. di Flora, S. Russo, and K. Raatikainen, “Unearthing design patterns to support context-awareness,” in Pervasive Computing and Communications Workshops

Specification

