TU WIEN TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Improving System Integration Using a Modular Configuration Specification Language

Markus Raab

Vienna University of Technology

Institute of Computer Languages, Austria

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

Outline

- **Motivation**
  - Example
  - Goal
  - Problem
- **SpecElektra**
- **Evaluation**
  - Contributions
  - Benchmarks
- Conclusion

Elektra's Logo

Example

- Mobile Device
- Tracker software
- Low battery
  - reduce time synchronisation
- With standard software
  - `ntpd`
  - we need to modify configuration file `ntp.conf`

Goals

- Context-Aware

  e.g. battery status

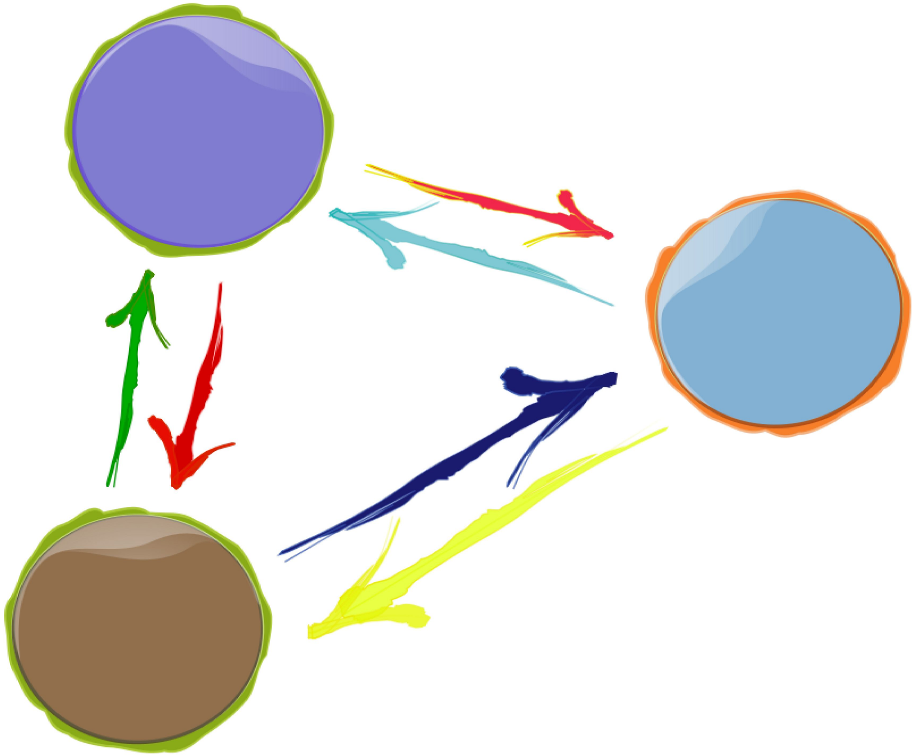
- Customizable

  adapt to user

- System-oriented Challenges

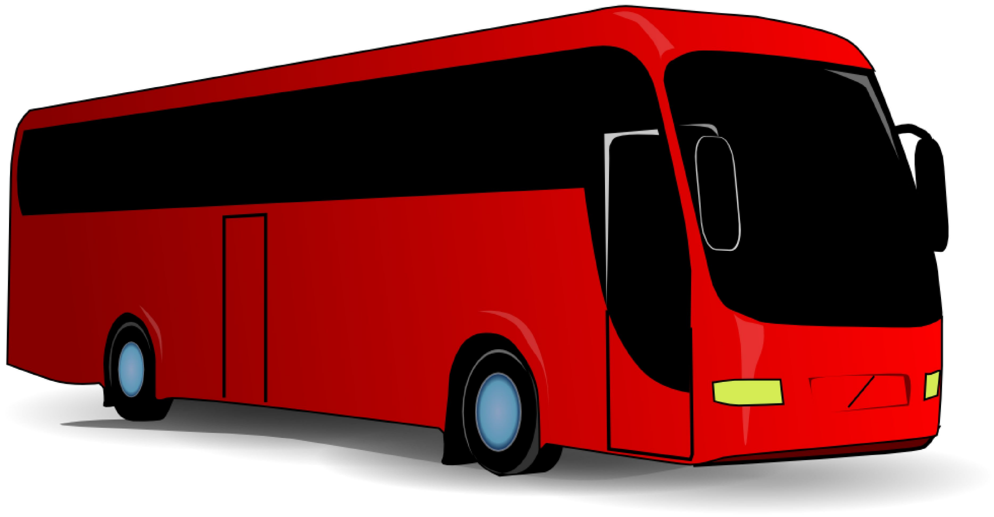
  changes across the stack

  changes in requirements



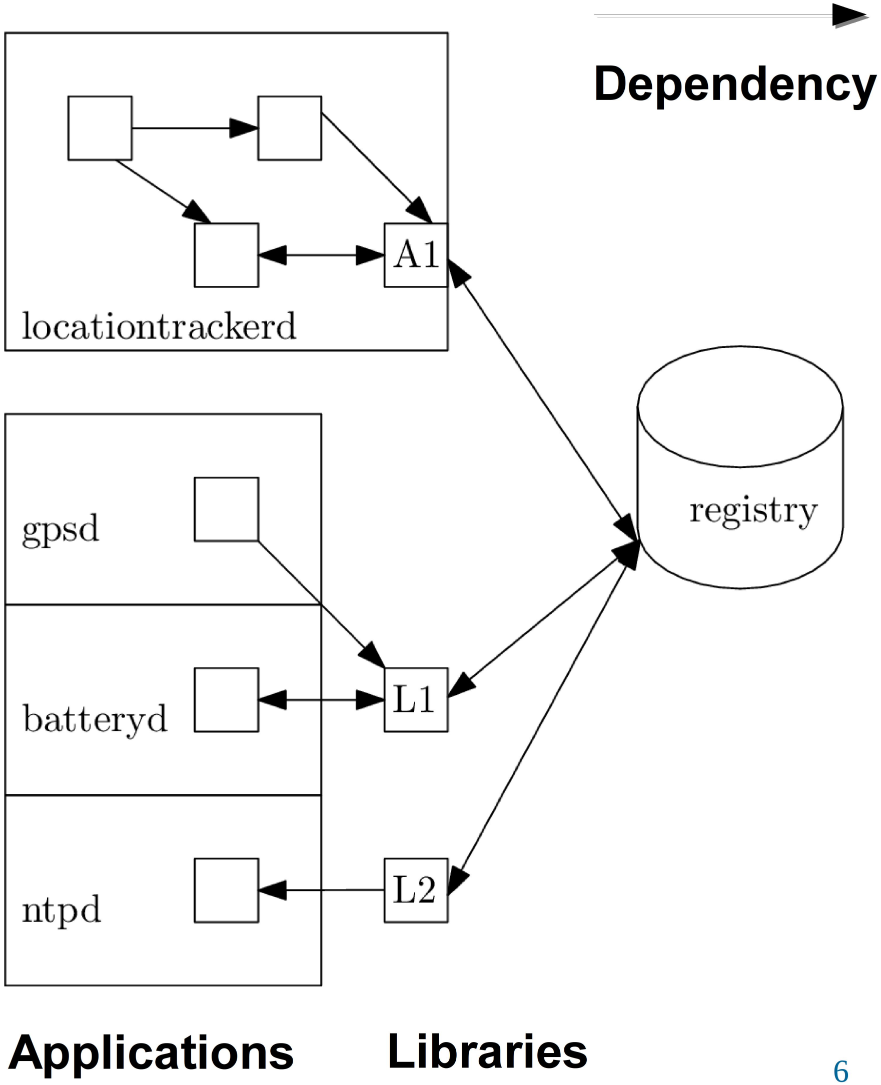
Example cont.

- System deployed in vehicle
- Reuse of same software
- Requirements change:
  - battery large enough
  - but higher resolution for higher speed needed



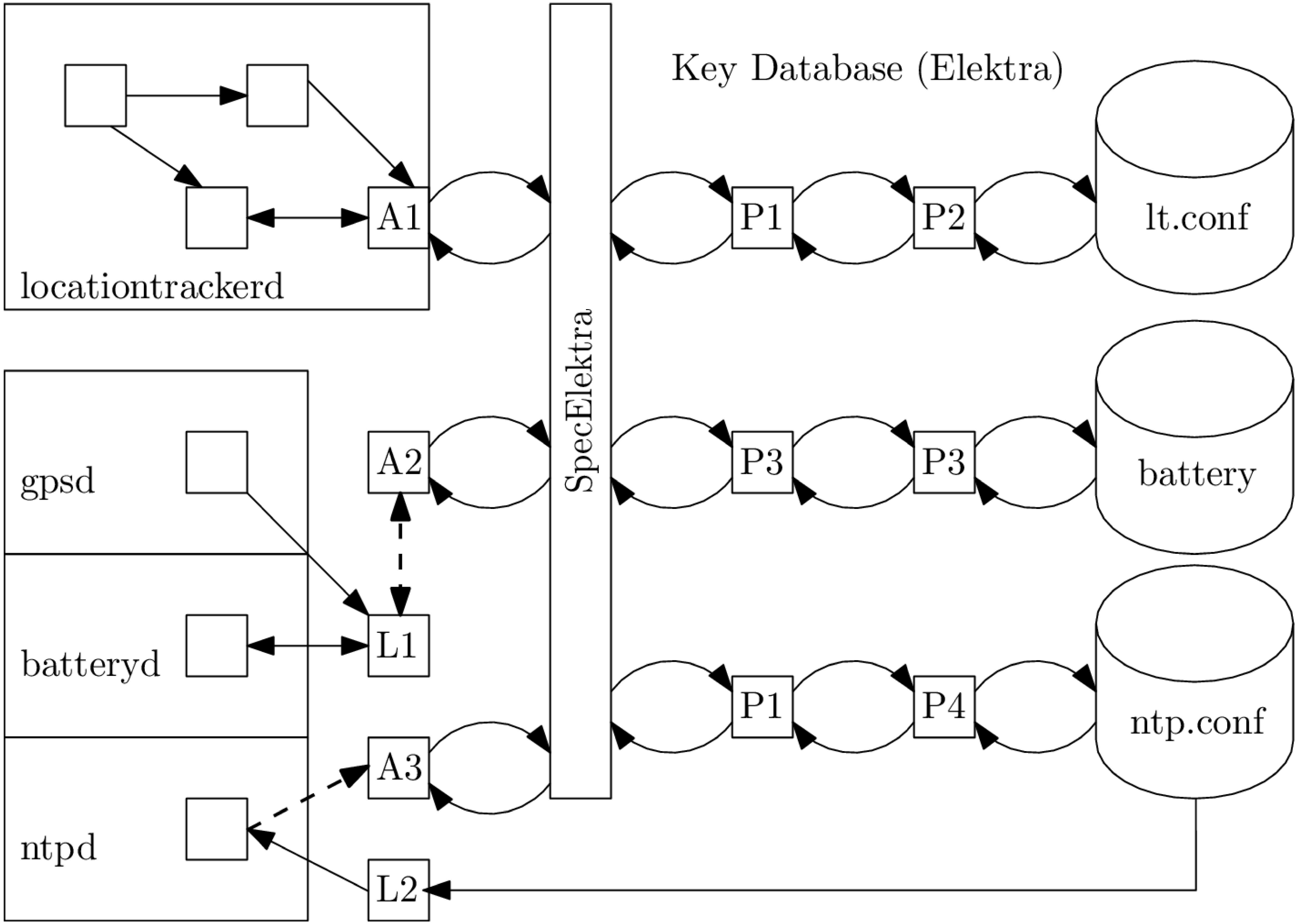
Possible Architecture

- Typically used
- Strong coupling
- Applications need to be adapted
- No application-specific behavior



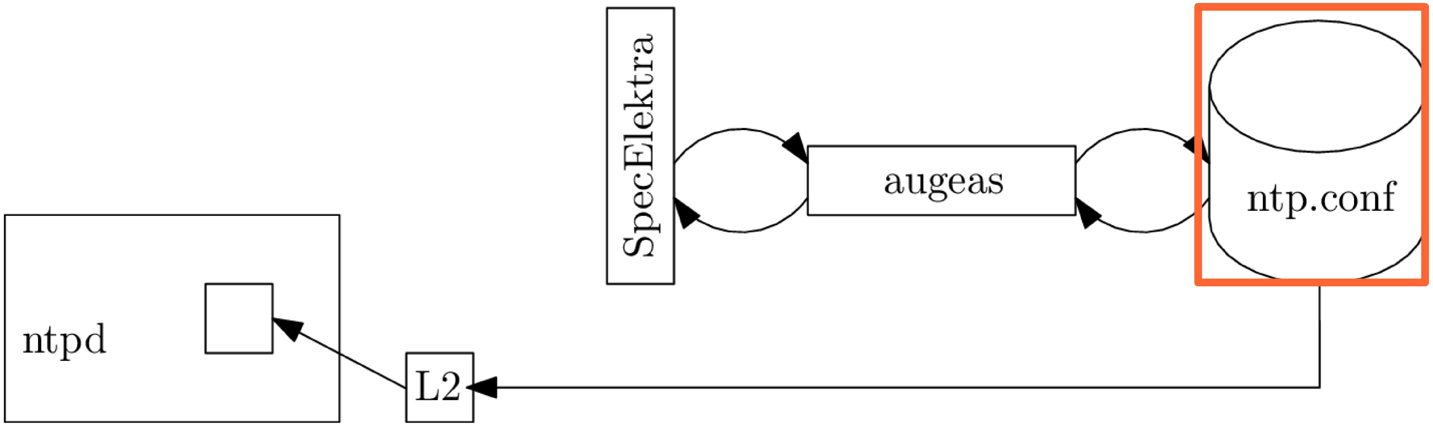
SpecElektra

Vertical and Horizontal Modularity



Ntpd

- Integration of existing configuration files



```
[ntp]
mountpoint = ntp.conf
infos/plugins = augeas
[ntp/maxpoll]
```

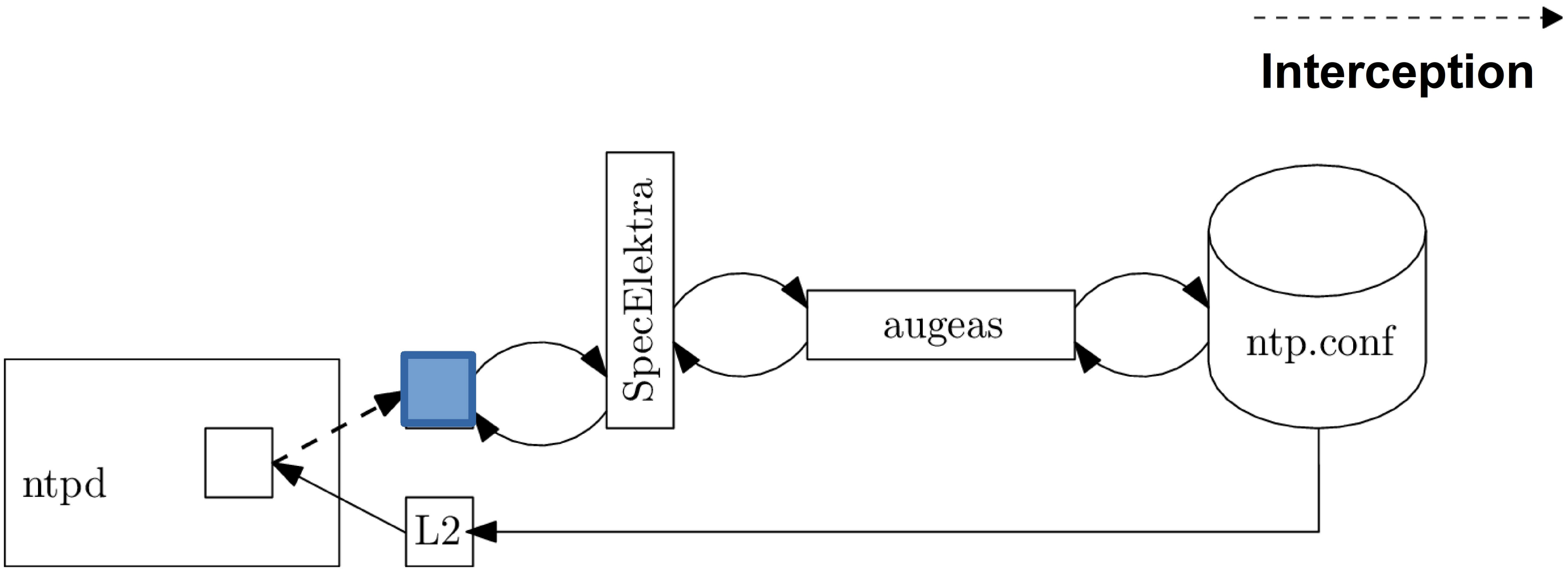
filtered by plugins

ntp.conf

```
server myntpsvr iburst burst minpoll 4 maxpoll 4
```

Ntpd

- Integration via getenv-adapter

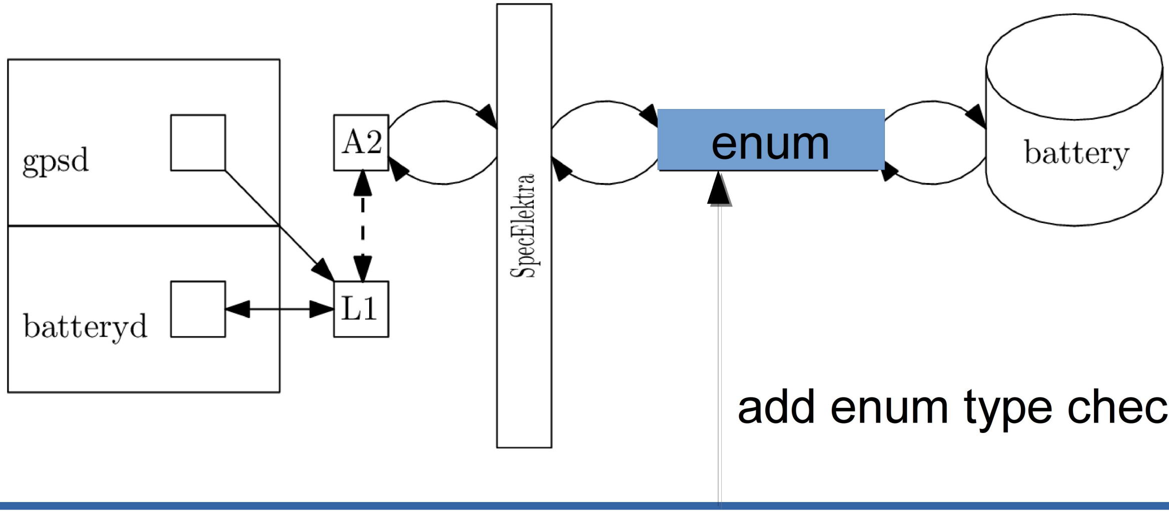


getenv("NTPD_UPDATEINTERVAL") → [env/NTPD_UPDATEINTERVAL]

- Advantage: Every access via plugins

Battery

- Type checks

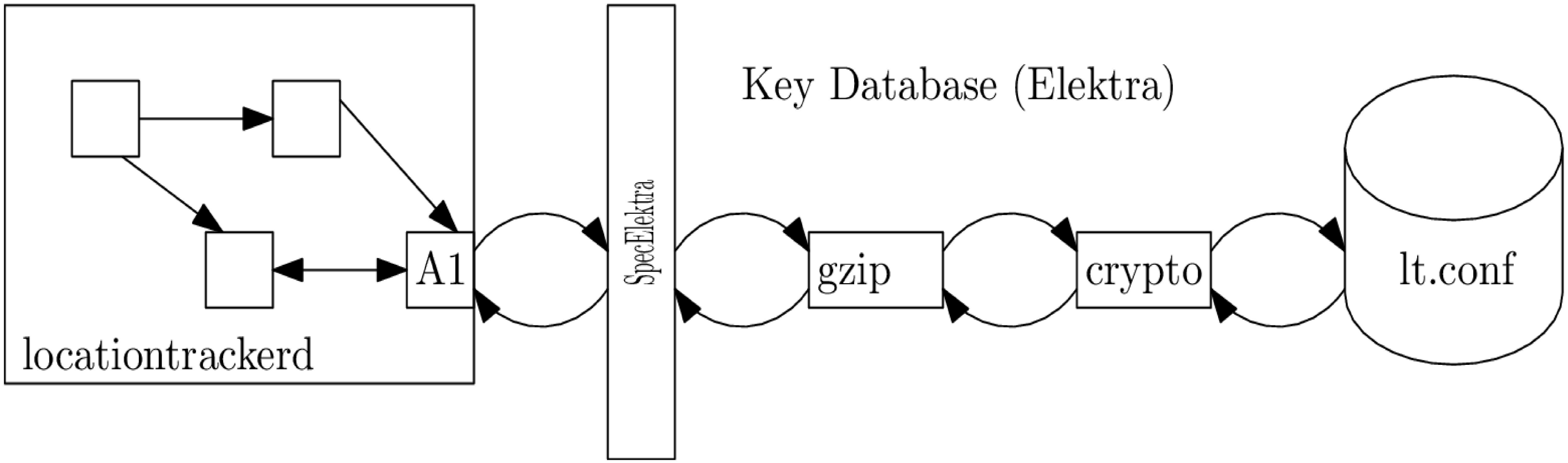


add enum type checker

```
[battery/level]
check/enum = 'critical', 'low', 'high', 'full'
```

Location Tracker

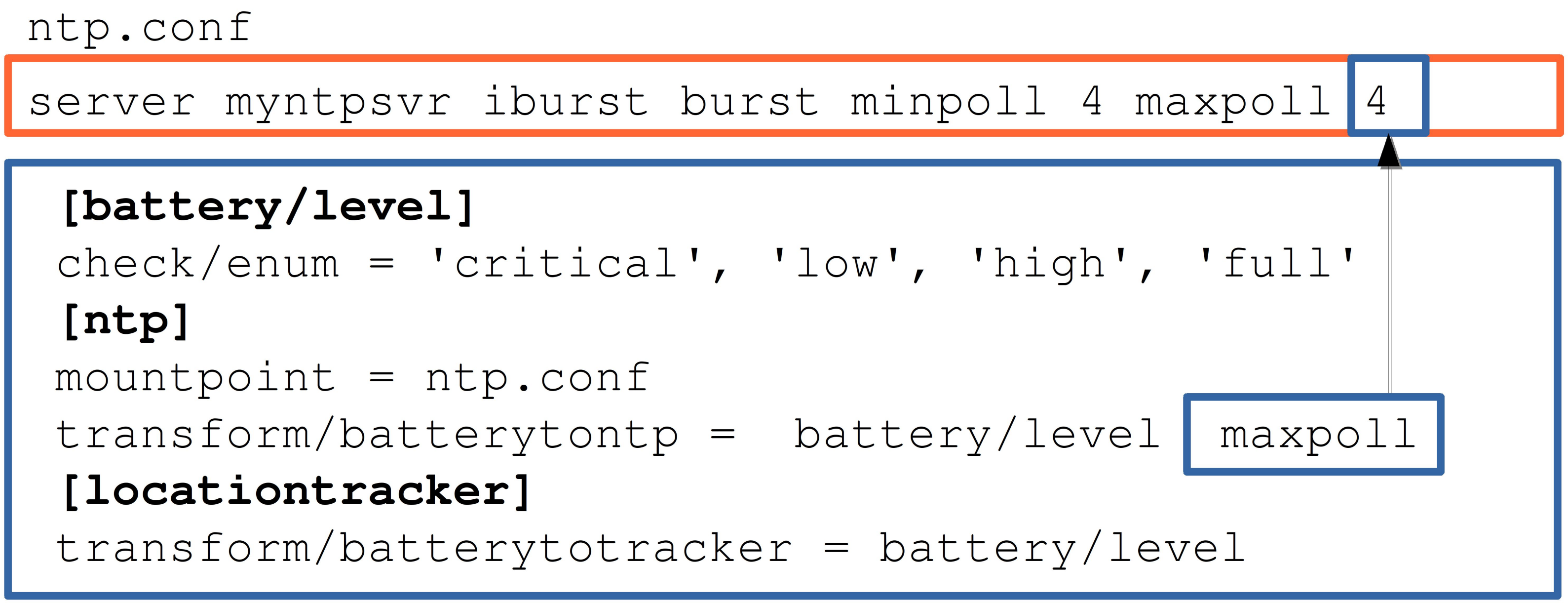
- Integration via code generator



```
[locationtracker/secret]
infos/plugins = gzip crypto
```

Specification

- Keys in **[ ]** refer to configuration
- Other lines are properties
- Properties specify configuration
- E.g. transformation properties



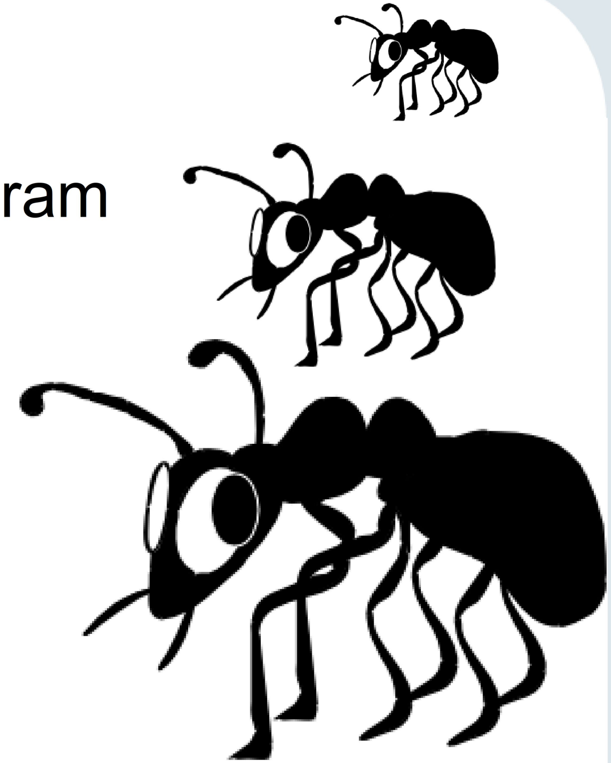
ntp.conf

```
server myntpsvr iburst burst minpoll 4 maxpoll 4
```

```
 [battery/level]
check/enum = 'critical', 'low', 'high', 'full'
 [ntp]
mountpoint = ntp.conf
transform/batterytontp =  battery/level  maxpoll
 [locationtracker]
transform/batterytotracker = battery/level
```

Evaluation

Generic Plugins

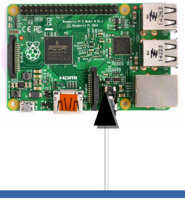


- (1) Configuration for plugin is a program
- (2) Compile time variability in plugin

```
#ifdef FEATURE
 …
#endif
```

- every variant is compiled
- tradeoff performance/flexibility
- transparently handled by properties in specification

High-Level Constraints



- Requirements from EEPROM:

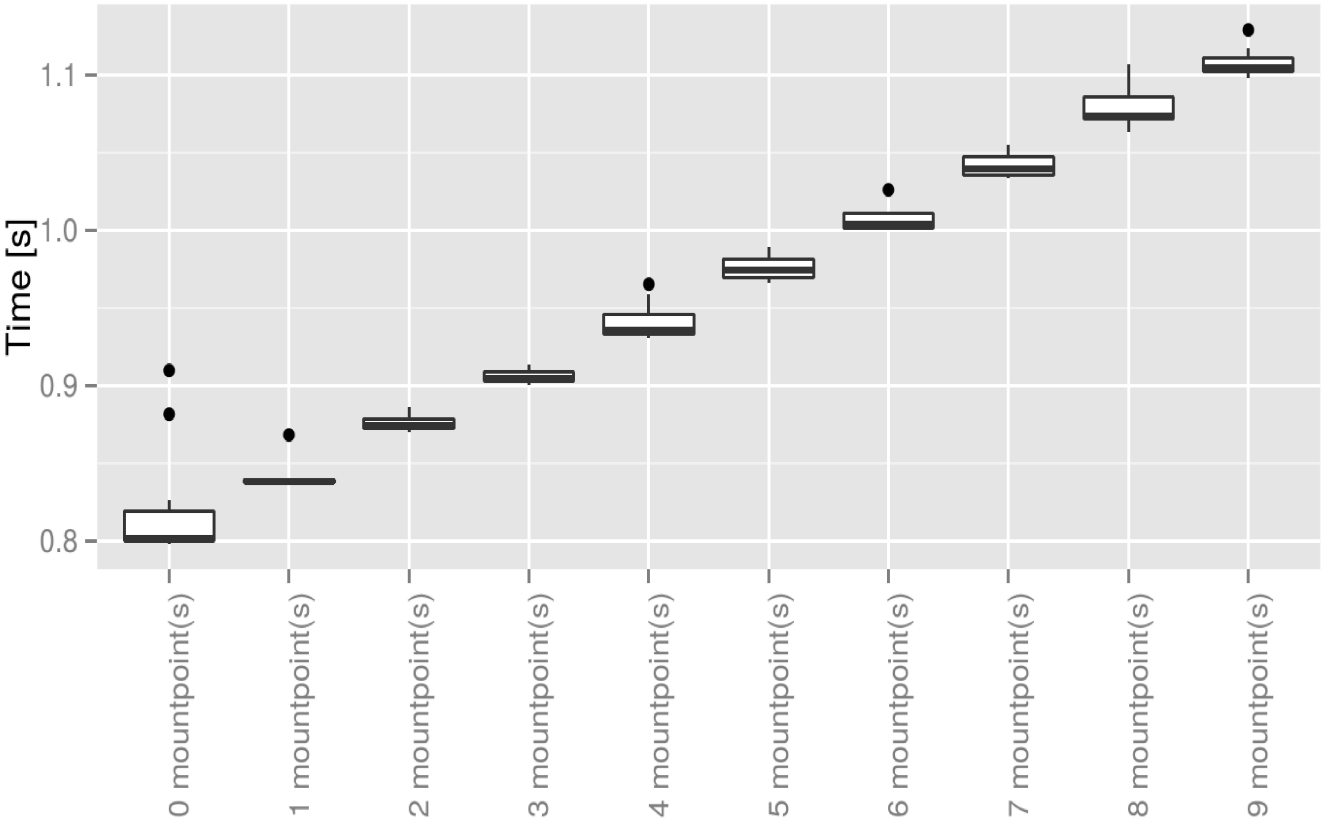
```
[device]
check/enum = 'wearable','smartphone','vehicle'
```

- Easily adaptable high-level constraints

```
[powersaving/gps]
assign/condition = (device != 'vehicle') ?
(battery/level) : ('full')
[gps/resolution]
assign/condition = (device == 'vehicle') ?
('high') : ('low')
```

Vertical Modularity

```
[benchmark/0]
mountpoint = /tmp/file0
[benchmark/1]
mountpoint = /tmp/file1
...
```



Horizontal Modularity

```
 [benchmark]
mountpoint = /tmp/file
infos/needs = iterate#0 iterate#1 ...
```

- increase number of plugins
- parsing files is dominant
- no overhead could be measured
- no reason to avoid modularity

Source Code

- Source Code released as free software within Elektra
  - >50 predefined plugins
  - support for hundreds kinds of configuration files
  - integrate standard software
  - specification is configuration (e.g. in XML, JSON)
- http://www.libelektra.org
  - version 0.8.15 released at 2016-02-16

Conclusion

- Vertical and Horizontal Modularity
  - observations and improvements
- Configuration Specification
  - validates and documents configuration
  - adapts behaviour of configuration access
  - cross-cutting concerns
  - high-level options for requirements
- Evaluation
  - acceptable overhead to improve vertical modularity
  - no measureable overhead for simple plugins

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

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

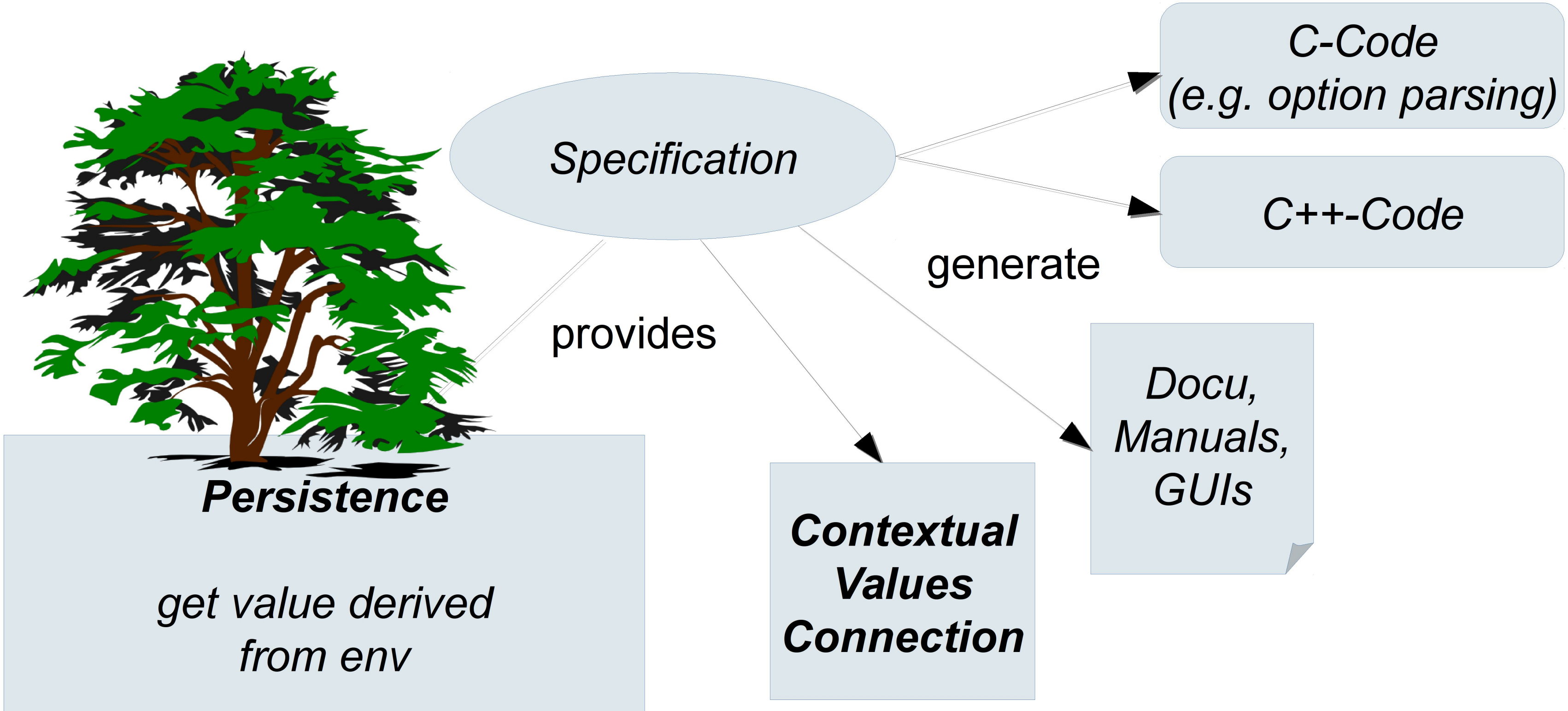
context variables (check on every usage)
M. von Löwis, M. Denker, and O. Nierstrasz, “Context-oriented programming: Beyond layers,” in Proceedings of the 2007 International Conference on Dynamic Languages

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



Specification

provides

generate

C-Code
(e.g. option parsing)

C++-Code

Docu,
Manuals,
GUIs

Contextual Values Connection

Persistence

get value derived from env

- *Runtime Checks*
- *Validation*
- *Upgrades*
- *Integration*

```
/%/%/%/person/greeting=Hi!
/German/%/%/person/greeting=Guten Tag!
/German/Austria/%/person/greeting=Servus!
/German/Austria/t/person/greeting=Griaß enk!
```