L03 Configuration Integration

Markus Raab

Institute of Information Systems Engineering, TU Wien

24.03.2021

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

Configuration Libraries

1. Configuration Libraries
2. Lightweight to Strong Integration
3. Sharing Configuration
4. Meeting
   - Recapitulation
   - Assignments
   - Preview

Learning Outcomes

Students will be able to

- remember strategies for configuration integration.

Configuration Access APIs

An ***application programming interface (API)*** defines boundaries on source code level. Better APIs make the execution environment easier and more uniformly accessible.

Configuration access is the part of every software system concerned with fetching and storing configuration settings from and to the execution environment. There are many ways to access configuration [1, 2, 4]. ***Configuration access APIs*** are APIs that enable configuration access.

Configuration Access APIs

- `char * getenv (const char * key)`

Configuration Access APIs

- `char * getenv (const char * key)`
- `ConfigStatus xf86HandleConfigFile(Bool autoconf)`

Configuration Access APIs

- `char * getenv (const char * key)`
- `ConfigStatus xf86HandleConfigFile(Bool autoconf)`
- `long pathconf (const char *path, int name)`

Configuration Access APIs

- `char * getenv (const char * key)`
- `ConfigStatus xf86HandleConfigFile(Bool autoconf)`
- `long pathconf (const char *path, int name)`
- `long sysconf (int name)`

Configuration Access APIs

- `char * getenv (const char * key)`
- `ConfigStatus xf86HandleConfigFile(Bool autoconf)`
- `long pathconf (const char *path, int name)`
- `long sysconf (int name)`
- `size_t confstr (int name, char *buf, size_t len)`

Configuration Access Points

Within the source code the ***configuration access points*** are configuration access API invocations that return configuration values.

```
1 int main()
2 {
3     getenv ("PATH");
4 }
```

Configuration Libraries

Configuration libraries provide implementations for a configuration access API.
Trends:

- flexibility to configure configuration access (e.g., `https://commons.apache.org/proper/commons-configuration/`)

Configuration Libraries

Configuration libraries provide implementations for a configuration access API.
Trends:

- flexibility to configure configuration access (e.g., `https://commons.apache.org/proper/commons-configuration/`)
- more type safety (e.g., `http://owner.aeonbits.org/`, code generation)

Configuration Libraries

Configuration libraries provide implementations for a configuration access API.
Trends:

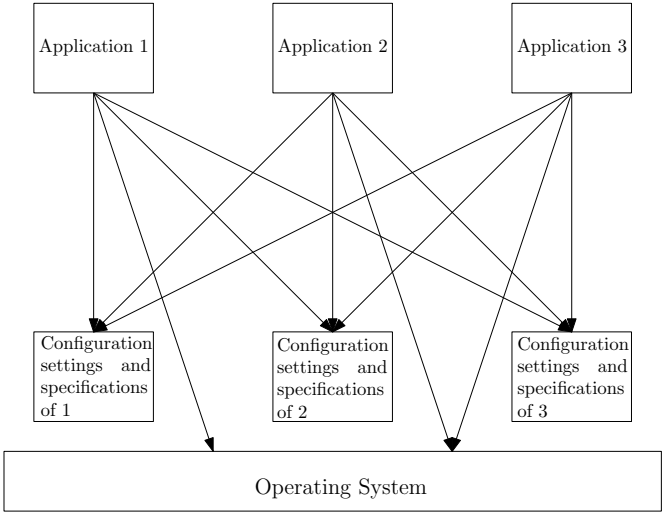
- flexibility to configure configuration access (e.g., `https://commons.apache.org/proper/commons-configuration/`)
- more type safety (e.g., `http://owner.aeonbits.org/`, code generation)
- specifications and introspection (gsettings, XML/JSON, Elektra)

Configuration Libraries

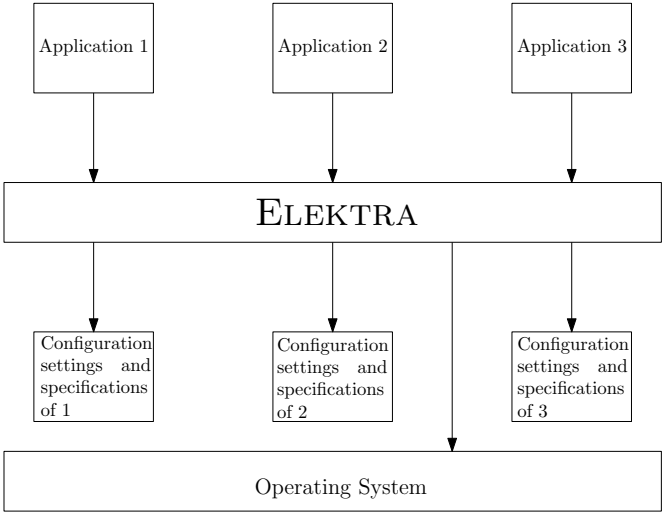
Configuration libraries provide implementations for a configuration access API.
Trends:

- flexibility to configure configuration access (e.g., `https://commons.apache.org/proper/commons-configuration/`)
- more type safety (e.g., `http://owner.aeonbits.org/`, code generation)
- specifications and introspection (gsettings, XML/JSON, Elektra)
- configuration integration (UCI, Augeas, Elektra)

Current Situation



Wanted Situation



L03 Configuration Integration

Markus Raab

Institute of Information Systems Engineering, TU Wien

24.03.2021

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

Lightweight to Strong Integration

1. Configuration Libraries
2. Lightweight to Strong Integration
3. Sharing Configuration
4. Meeting
   - Recapitulation
   - Assignments
   - Preview

Lightweight Integration

Specify already-existing configuration files:

```
1 [ntp]
2   mountpoint:=ntp.conf
3   infos/plugins:=ntp
```

Works well for configuration management tools.

Medium Integration

Having frontends that implement existing **APIs** decouple applications from each other. These applications continue to use their specific configuration accesses, but ELEKTRA redirects their configuration accesses to the shared key database.
Possible APIs:

- `getenv`

Medium Integration

Having frontends that implement existing **APIs** decouple applications from each other. These applications continue to use their specific configuration accesses, but ELEKTRA redirects their configuration accesses to the shared key database.
Possible APIs:

- `getenv`
- open/close of configuration files

Medium Integration

Having frontends that implement existing **APIs** decouple applications from each other. These applications continue to use their specific configuration accesses, but ELEKTRA redirects their configuration accesses to the shared key database.
Possible APIs:

- `getenv`
- open/close of configuration files

Also needs application-specific specifications.

Strong Integration

Change the application so that it directly uses Elektra.
Advantages:

- Elektra's features always available

Strong Integration

Change the application so that it directly uses Elektra.
Advantages:

- Elektra's features always available
- more type safety

Strong Integration

Change the application so that it directly uses Elektra.
Advantages:

- Elektra's features always available
- more type safety
- administrators can choose configuration file formats

Strong Integration

Change the application so that it directly uses Elektra.
Advantages:

- Elektra's features always available
- more type safety
- administrators can choose configuration file formats
- notification and logging

Strong Integration

Change the application so that it directly uses Elektra.
Advantages:

- Elektra's features always available
- more type safety
- administrators can choose configuration file formats
- notification and logging
- only one parser involved

Strong Integration

Change the application so that it directly uses Elektra.
Advantages:

- Elektra's features always available
- more type safety
- administrators can choose configuration file formats
- notification and logging
- only one parser involved
- no specification for binding needed

Strong Integration

Change the application so that it directly uses Elektra.
Advantages:

- Elektra's features always available
- more type safety
- administrators can choose configuration file formats
- notification and logging
- only one parser involved
- no specification for binding needed
- no built-in defaults: everything is introspectable

Strong Integration

Different implementations strategies:

- have some application-specific API which uses `KeySet`

Strong Integration

Different implementations strategies:

- have some application-specific API which uses `KeySet`
- use one of KeySet's language bindings

Strong Integration

Different implementations strategies:

- have some application-specific API which uses `KeySet`
- use one of KeySet's language bindings
- use Elektra's high-level API (currently only C)

Strong Integration

Different implementations strategies:

- have some application-specific API which uses `KeySet`
- use one of KeySet's language bindings
- use Elektra's high-level API (currently only C)
- use code generation

Strong Integration

Examples:

- LCDproc
- Oyranos
- for GNOME: `gsettings`
- for KDE: `kconfig`

L03 Configuration Integration

Markus Raab

Institute of Information Systems Engineering, TU Wien

24.03.2021

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

Sharing Configuration

1. Configuration Libraries
2. Lightweight to Strong Integration
3. Sharing Configuration
4. Meeting
   - Recapitulation
   - Assignments
   - Preview

- idea: make default values better
- generalization of sharing configuration values
- examples: language settings, default printer, ...

- idea: make default values better
- generalization of sharing configuration values
- examples: language settings, default printer, ...

Can be derived from:

- other configuration settings (override/fallback)
- context [3]
- hardware/system (problem with dependences)

- idea: make default values better
- generalization of sharing configuration values
- examples: language settings, default printer, ...

Can be derived from:

- other configuration settings (override/fallback)
- context [3]
- hardware/system (problem with dependences)

- XServer vs. gpsd

Examples

Context:

```
1 [slapd/threads/listener]
2 context :=/slapd/threads/%cpu%/listener
```

Examples

Context:

```
1 [slapd/threads/listener]
2 context:=/slapd/threads/%cpu%/listener
```

Calculation with conditionals plugin
(e.g., switch off GPS if battery low):

```
1 [gps/status]
2 assign:=(battery > 'low') ? ('on') : ('off')
```

L03 Configuration Integration

Markus Raab

Institute of Information Systems Engineering, TU Wien

24.03.2021

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

Meeting

1 Configuration Libraries

2 Lightweight to Strong Integration

3 Sharing Configuration

4 Meeting
- Recapitulation
- Assignments
- Preview

Learning Outcomes

Students will be able to

- remember strategies for configuration integration.

Task

Which configuration access APIs do you know?
What are the differences between these APIs?

Question

Which forms of configuration integration exist?

Answer

- Lightweight Integration: Mount
- Medium Integration: Specify
- Strong Integration: Modify

Question

Which forms of configuration integration exist?

Answer

- Lightweight Integration: Mount existing configuration files for CM tools
- Medium Integration: Specify
- Strong Integration: Modify

Question

Which forms of configuration integration exist?

Answer

- Lightweight Integration: Mount existing configuration files for CM tools
- Medium Integration: Specify how to access via existing APIs
- Strong Integration: Modify

Question

Which forms of configuration integration exist?

Answer

- Lightweight Integration: Mount existing configuration files for CM tools
- Medium Integration: Specify how to access via existing APIs
- Strong Integration: Modify the application to use a configuration library that has support for configuration specifications.

Question

Given a strong integration, how can we reuse the same configuration setting for different applications?

Question

Given a strong integration, how can we reuse the same configuration setting for different applications?

Answer

- Implement support directly in application to fetch setting from central location.
- Override/fallback links in specification.
- Calculate/transform values in specification.
- Use CM code to copy the settings to all places as needed.

H1

Task

What do you want to do as homework?

P0

Task

Do you already have a suitable project?

T0 Issues

Task

Did you already find enough suitable issues?

Develop with Elektra

Task

Can you already compile Elektra and software using Elektra?

L04: Sources of Configuration

- Configuration file formats
- Environment variables
- Command-line arguments
- Abstractions
- Complexity

[1] 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.

[2] Emre Kiciman and Yi-Min Wang. Discovering correctness constraints for self-management of system configuration. In *International Conference on Autonomic Computing, 2004. Proceedings.*, pages 28–35. IEEE, May 2004. doi: 10.1109/ICAC.2004.1301344.

[3] 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.

[4] 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.