Recapitulation
0000000

Terms and Properties
00000

Validation
000000000000

Configuration Management

Markus Raab

Institute of Information Systems Engineering, TU Wien

5.6.2019

Recapitulation
0000000

Terms and Properties
00000

Validation
00000000000

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 Libary
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 and register for exam
26.06.2019: exam

Recapitulation
0000000

Terms and Properties
00000

Validation
000000000000

Tasks for today

(until 05.06.2019 23:59)

Task

Submit teamwork and homework.

Please also register for the exam in TISS.

Recapitulation
ooooooo

Terms and Properties
ooooo

Validation
oooooooooooo

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

Recapitulation
○○○○○○○

Terms and Properties
○○○○○

Validation
○○○○○○○○○○○○○

Recapitulation
ooooooo

Terms and Properties
ooooo

Validation
oooooooooooo
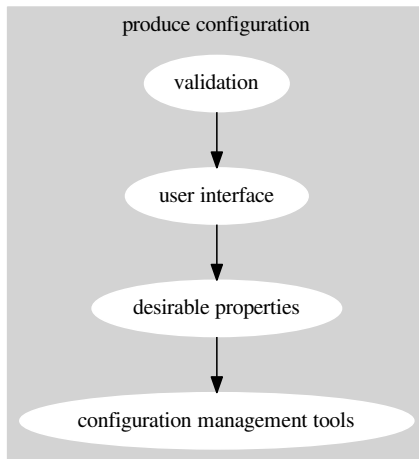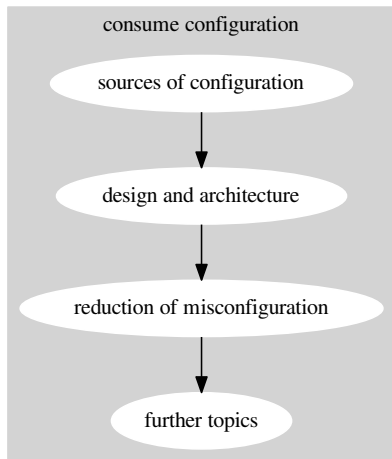
Recapitulation

1. Recapitulation

2. Terms and Properties

3. Validation

Early detection (Recapitulation)

> **Task**
>
> When do we want to detect misconfiguration?

Phases when we can detect misconfigurations:

- Compilation stage in configuration management tool
- Writing configuration settings on nodes
- Starting applications (load-time)
- When configuration setting is actually used (run-time)

> **Problem**
>
> Earlier versus more context.

Recapitulation
○●○○○○○

Terms and Properties
○○○○○

Validation
○○○○○○○○○○○○

Contextual Values (Recapitulation)

Task

What are contextual values?

Tanter [7] 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"* [7]. Side effects are limited to the respective context [4].

Definition Context (Recapitulation)

> **Task**
>
> What is context-aware configuration?

As adapted from Chalmers [1]:

> **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.

Recapitulation
○○○●○○○

Terms and Properties
○○○○○

Validation
○○○○○○○○○○○○○

Elektra (Recapitulation)

- 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 settings

Definition Configuration Management (Recapitulation)

- 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.
- has means to describe the desired configuration of the whole managed system.
- ensures that the execution environment of installed applications is as required.

Recapitulation
○○○○○○●○

Terms and Properties
○○○○○

Validation
○○○○○○○○○○○○○

Possible Benefits of CM (Recapitulation)

- All advantages scripts have:
  Documentation, Customization, Reproducability
- Declarative description of the system
  (Infrastructure as Code [2])
- Auditability
- Less configuration drift
- Error handling
- Pull/Push
- Reusability
- (Resource) Abstractions

Recapitulation
○○○○○○○●

Terms and Properties
○○○○○

Validation
○○○○○○○○○○○○○

Learning Outcomes

Students will be able to

- define terms of properties of CM.
- analyze validation specifications.
- remember the difference between checking the specifications vs. checking the settings.

Recapitulation
0000000

Terms and Properties
00000

Validation
000000000000

Terms and Properties

1. Recapitulation

2. Terms and Properties

3. Validation

Recapitulation
○○○○○○○

Terms and Properties
●○○○○

Validation
○○○○○○○○○○○○○

Infrastructure as Code

Once we described configuration settings, configuration settings are simply an instantiation of the configuration specifications.

Code describing the instantiation is **CM code**.

Auditability: Being informed about status and changes in the infrastructure.

> ### Goal
> Single Source of Truth

Recapitulation
0000000

Terms and Properties
○●○○○

Validation
000000000000

Configuration Drift

Are derivations of the "Single Source of Truth" (the CM code).
Caused by:

- manual configuration changes by administrators
- manual configuration changes by end users
- differences in updates (e.g., skipped or failed updates)
- failed attempts to change configuration
- applying different versions of CM code
- . . .

Push vs. Pull

- Push is more interactive.
- Push cannot do its job if nodes are not reachable.
- Push needs additional techniques to scale with many nodes.
- Push demands access to servers from a single server.
- Pull needs additional monitoring to know when a patch has been applied.
- Pull needs resources even if nothing is to do.

Task
Do you prefer push or pull? What does your CM tool of choice use?

Recapitulation
0000000

Terms and Properties
000●0

Validation
000000000000

Idempotence

idem + potence (same + power)

Yield same result with any number of applications ($n \geq 1$):

$$f(f(x)) = f(x)$$

Recapitulation
0000000

Terms and Properties
0000●

Validation
000000000000

Siméon and Wadler [6] describe two further properties:

Self-describing means that from the configuration file alone we are
able to derive the correct data structure [6].

Round-tripping means that if a data structure is serialized and then
parsed again, we end up with an identical data
structure [6].

The data structure could be a KeySet.

Round-tripping is a prerequisite of idempotence.

Task

Explain the three concepts your neighbor (idempotence,
self-describing, round-tripping).

Recapitulation
0000000

Terms and Properties
00000

Validation
000000000000

Validation

1 Recapitulation

2 Terms and Properties

3 Validation

Recapitulation
0000000

Terms and Properties
00000

Validation
●000000000000

Checking Configurations

Following properties of configuration settings can be checked:

- values (data types)
- structure
- constraints
- semantic checks (e.g., IP, folder)
- domain-specific checks (e.g., databases)
- requirements (suitable configurations)
- context (context-aware configurations)

Recapitulation
0000000

Terms and Properties
00000

Validation
00●000000000

Elektra supports many other data types, each implemented in its own plugin(s):

check/type allows us to specify CORBA data types. Checking "any" (default) is always successful. The record and enum types defined by CORBA are not part of this plugin but of others as explained below.

check/enum supports a list of supported values denoted by array indexes.

check/bool transforms specific strings, for example "true" and "false", into the canonical boolean representation, i. e., "0" and "1".

check/ipaddr checks if a string is a valid IP address.

check/path checks presence, permissions, and type of paths in the file system.

check/date supports to check date formats such as POSIX, ISO8601, and RFC2822.

check/validation checks the configuration value with regular expressions.

check/condition checks using conditionals and comparisons.

check/math checks using mathematical expressions.

check/range allows us to check if numerical values are within a range.

trigger/error allows us to express unconditional failures.

check/rgbcolor allows us to check for RGB colors, by Philipp Gackstatter.

check/macaddr (will allow us to check for MAC addresses, by Thomas Bretterbauer)

Checking Specifications

The goals of checking SpecElektra are:

- Defaults must be present for safe lookups. This goal also implies that there must be at least one valid configuration setting.
- Types of default values must be compatible with the types of the keys.
- Every contextual interpretation of a key must yield a compatible type.
- Links must not refer to each other in cycles.
- Every link and the pointee must have compatible types.

Recapitulation
0000000

Terms and Properties
00000

Validation
00000●0000000

Example

```
1  [sw/org/abc/has_true_arg]
2     type := boolean
3     default := 0
4     override/#0 := /sw/org/abc/arg0
5     override/#1 := /sw/org/abc/arg1
```

Recapitulation
0000000

Terms and Properties
00000

Validation
000000●000000

Logfile Example

```
1 [slapd/logfile]
2    check/path:=file
```

Recapitulation
0000000

Terms and Properties
00000

Validation
000000●00000

Logfile Extensions

```
1 [slapd/logfile]
2   check/path := file
3   check/validation := ^/var/log/
4   check/validation/message := Policy violation:
5       log files must be below /var/log
```

Recapitulation
0000000

Terms and Properties
00000

Validation
00000000●0000

Error Messages

Error messages are extremely important as they are the main communication channel to system administrators.
Example specification:

```
1 [a]
2    check/type := long
3 [b]
4    check/type := long
5 [c]
6    check/range := 0-10
7    assign/math := ../a+../b
```

Recapitulation
0000000

Terms and Properties
00000

Validation
0000000000●0000

Error Messages

Problems:

- Generic vs. specific plugins
- General principles of good error messages [3]
- Give context
- Precisely locate the cause:

```
1 a=5    ; unmodified
2 b=10   ; modification bit in metadata
3        ; is only set here
4 c=15   ; unmodified by user but changed
5        ; later by assign/math
```

Example Error Message

```
Sorry, I was unable to change the configuration settings!
Description: I tried to set a value outside the range!
Reason: I tried to modify b to be 10 but this caused c to
        be outside of the allowed range (0-10).
Module: range
At: sourcefile.c:1234
Mountpoint: /test
Configfile: /etc/testfile.conf
```

Recapitulation
0000000

Terms and Properties
00000

Validation
0000000000000

Conclusion

- Definition and challenges in configuration management.
- Properties: self-describing, idempotent, round-tripping.
- Validation is combined effort of devs and admins.

Recapitulation
0000000

Terms and Properties
00000

Validation
0000000000000●

Outlook

- Error Messages
- Configuration Management Tools and Languages
- User Interface: View of Administrators

[1] Daniel Chalmers. *Contextual mediation to support ubiquitous computing*. PhD thesis, University of London, 2002.

[2] Waldemar Hummer, Florian Rosenberg, Fábio Oliveira, and Tamar Eilam. Testing idempotence for infrastructure as code. In David Eyers and Karsten Schwan, editors, *Middleware 2013*, pages 368–388, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-45065-5.

[3] Michael J. Lee and Andrew J. Ko. Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the Seventh International Workshop on Computing Education Research*, ICER '11, pages 109–116, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0829-8. doi: 10.1145/2016911.2016934. URL http://dx.doi.org/10.1145/2016911.2016934.

[4] 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.

[5] 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.

[6] Jérôme Siméon and Philip Wadler. The essence of xml. pages 1–13, 2003. doi: 10.1145/604131.604132. URL http://dx.doi.org/10.1145/604131.604132.

[7] 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.