FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Error Injection in Specification-Based Configurations

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Michael Zronek, M.Sc.
Matrikelnummer 0951864

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr. Franz Puntigam
Mitwirkung: Dr.techn. Dipl.-Ing. Markus Raab

Wien, 10. Jänner 2020

Michael Zronek | Franz Puntigam

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Error Injection in Specification-Based Configurations

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Michael Zronek, M.Sc.
Registration Number 0951864

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr. Franz Puntigam
Assistance: Dr.techn. Dipl.-Ing. Markus Raab

Vienna, 10th January, 2020

Michael Zronek   Franz Puntigam

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Michael Zronek, M.Sc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. Jänner 2020

Michael Zronek

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Danksagung

In etwas unkonventioneller Manier möchte ich meinem verstorbenem Hund Pia danken, welcher mir in schlechten Zeiten die meiste Kraft und Trost gespendet hat.

Weiters gebührt auch meinen Eltern ein großer Dank, welche mich finanziell enorm unterstützt haben. Ich schätze es sehr und weiß, dass so etwas nicht selbstverständlich ist.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

In an unconventional way, I want to thank my deceased dog Pia, which gave me the most power and consolation in bad times.

Furthermore I want to thank my parents for greatly supporting this study financially. I appreciate their help very much and do not take such support as granted.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Spezifikationen im Kontext von Konfigurationen sind eine Möglichkeit Konfigurationswerte einzuschränken. Mit diesen Spezifikationen kann man Fehler, welche Benutzer oder Administratoren machen, limitieren.

Das Schreiben dieser Spezifikationen benötigt allerdings Ressourcen wie unter anderem einen Zeitaufwand. Es ist unklar wie effektiv Spezifikationen sind und wie viele Konfigurationsfehler sie abfangen. Ein weiterer wesentlicher Aspekt sind Fehlermeldungen, welche eine Verbesserung zur applikations-nativen Fehlermeldung sein sollten.

Diese Arbeit beschäftigt sich damit, wie aufwändig es ist eine Spezifikation für LCDproc und Cassandra zu schreiben. Beide Applikationen haben eine mittlere Größe mit 190-300 Konfigurationswerten. Wir haben den Zeitaufwand mitgeschrieben, welchen es braucht, um diese Spezifikationen zu schreiben. Wir haben auch das Tool Lyrebird geschrieben, welches automatische Fehler in diese Applikationen injiziert. Mit Lyrebird können wir die Anzahl an Fehlkonfigurationen messen, welche mit und ohne Spezifikation erkannt wurden. In weiterer Folge werden dann mittels manueller Analyse die Fehlermeldungen auf 5 wesentliche qualitative Aspekte begutachtet.

Unser Ergebnis zeigt, dass der extra Layer einer Konfigurationsspezifikation bei der Fehlererkennung variierenden Erfolg hat. Je komplexer die Applikation ist, desto weniger wahrscheinlich wird die Spezifikation komplizierte Fehler abfangen. Die Qualität der Fehlermeldungen ist allerdings wesentlich besser, bedingt durch ein einheitliches und standardisiertes Format.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Specifications in the context of configurations are ways to describe and constrain configuration values. These specifications can be used to limit the possible errors users and admins can do.

Writing these specifications though requires resources such as time. It is also unclear how effective such specifications are and how many misconfigurations they can catch. The quality of the error message is also a point of concern because it should be an improvement to the application's native configuration validation.

Our thesis will investigate how effortful it is to write specifications for LCDproc and Cassandra, both medium sized applications with 190-300 configuration settings. We tracked the time needed for writing specifications for both applications and we developed the tool Lyrebird for automatic error injection. With Lyrebird we check how many misconfigurations these applications catch with and without a specification. The error messages are then manually analyzed for 5 key features in terms of quality.

Our results show that the additional layer of a configuration specification varies in the detection rate of errors. The more complex an application is, the less likely a specification will catch difficult errors. The quality of the error messages though perform significantly better due to an uniform and standardized error message.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**

2 State of the Art **5**
2.1 Configuration Specification 5
2.2 Error Injection 6
2.3 Error Messages 9

3 Methodology **11**
3.1 Error Types 12
3.2 Automatic Invalid Configuration Setting Injection - Lyrebird 14

4 Evaluation **21**
4.1 Evaluated Applications 21
4.2 LCDproc Results 23
4.3 Cassandra Results 31
4.4 Plugin Development 37
4.5 Discussion 38
4.6 Implications - Improving Error Messages 40

5 Conclusion **43**

List of Figures **45**

List of Tables **47**

Bibliography **49**

Other references **53**

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 1

Introduction

Nearly every modern software application requires a configuration solution before working as expected. Various configuration file formats within a single application is not uncommon and the number of configuration solutions constantly rose over the past decade [35][1, p. 68]. For example, Firefox had around 500 configuration settings in 2008 while having more than 2000 in 2016. Just a single invalid configuration setting (ICS) can lead to unexpected behavior or even crashes during runtime. Xu et al. has shown that many mature software systems are subject to latent configuration error [43].

Correcting the configuration error can be very costly and time-intensive. For instance, developers of Squid needed around 48 hours to patch the diskd_program configuration setting which caused a 100% CPU usage when trying to rotate a logfile. The problem occurred when one configuration setting in the configuration solution contained a non-existent path [43][2].

Even when finding the configuration settings which caused the configuration error, it can be a trial and error approach to fix the problem. Imagine a configuration setting which only supports values from an enumeration that are not (fully) documented.

To counteract the problem of setting an incorrect configuration setting, a specification can be a remedy. A specification describes the possible configuration values of a configuration setting in such a way that it constrains it. A typical example would be a configuration setting that expects a number and therefore restricts the use of strings. A specification also provides documentation as well as default values if the configuration setting is omitted. Another benefit is the point of feedback a specification can provide. While configuration settings are usually parsed and loaded during system startup (which can take even several minutes), the validation of a configuration value with a specification can be done right after changing it. The earlier the user receives a feedback about an ICS, the easier and less frustrating it is to solve the ICS.

Despite all these benefits it is time consuming to write a configuration specification. Depending on the case, it can either be straightforward, such as writing allowed ranges of number configuration settings, to very complex specifications like difficult regular expressions or even semantic checks such as a valid and reachable IP address. These checks themselves can be wrong too, leading to rejected configuration settings despite being valid. An example would be in LCDproc (a system-information display software) in which a configuration setting accepts a hexadecimal port range from `0x200 - 0x400`. The corresponding regular expression looks like this: 0x([2-3][0-9A-F]2|400) and is subject to errors such as the maximum allowed port of `0x400`.

In our thesis one major question we want to investigate if writing specifications for configurations is done in reasonable time and if those specifications are also valuable to the users and administrators. As of now we did not find any research literature which investigates the effort and problems it requires to write a specification. Additionally, if there are constraints which are currently not possible to check against by the specification language, it is of interest how much time it takes to extend the specification language. With the help of Elektra[14], a modular configuration framework with an integrated specification language, we want to remedy this open research area. Our thesis will answer the following question:

RQ1. How costly it is to write specifications for medium-sized applications? How effortful are improvements and application specific extensions to the specification language?

We tracked the time needed to write configuration specifications for applications with 190 to 300 configuration settings. We also tracked the time needed to write three Elektra extensions which allow us to enhance the specification language. With these records we will be able to answer RQ1.

One way a specification is valuable to the user is if it catches at least as many ICS as the application itself. In our thesis we want to investigate if this is actually the case because a specification might not have the same information as the application. Complex interdependent configuration settings are not easily handled by the specification, but can be caught by the application logic itself much easier. In the database Cassandra for example, users can implement certain Java classes by themselves and provide them through configuration settings. It is easier to verify that a certain interface is correctly implemented by the application who uses it, compared to an external validation tool.

The question if the application or specification performs better and catches more ICS is yet another open research area for which we could not find scientific literature. With the use of different error types (e.g., typos, off-by-one error, semantic errors, etc.) we will

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

answer the following question:

RQ2. How do specification-based configurations compare to non-specification-based configurations in medium-sized applications in regards of catching invalid configuration settings?

We wrote the application *Lyrebird* which is freely available on GitHub [47] to be able to compare the catch rate of applications to Elektra. *Lyrebird* injects ICS in configurations from applications and starts the application to see its system reaction. The same ICS are also injected into the configuration but with a specification written in the specification language `SpecElektra` and checked if the general configuration framework Elektra can catch more ICS. With *Lyrebird* we can quantify differences in error types and draw conclusions.

Another aspect of configuration specifications are error messages which they deliver. Bad error messages prolong the finding of the correct configuration solution. Sometimes the root cause is hard to detect and specifications should also take such problems into account. In the worst case the ICS is silently ignored, causing a potentially costly latent configuration error. For example, if in MySQL 5.5 an invalid path of the log-error configuration setting was given, a loss of all error logs occurred [33]. Our hypothesis is that in configurations with specifications, using an invalid configuration setting will give a useful error message which can provide the location, the wrong configuration setting and the reason why it is rejected (such as giving all allowed enumerations as in the example above).

We think that the extra layer of configuration validation can improve the quality of error messages. A uniform and consistent error message format for similar ICS errors better enables users to find the correct solution more quickly with less frustration. Therefore we want to answer the following question:

RQ3. How can error messages be improved with specification-based configurations?

We investigated literature for state-of-the-art error messages and found five key quality aspects of good error messages. We then manually went through all error messages that resulted from the ICS in the application and Elektra. Each message is analyzed for these quality attributes and both Elektra's as well as the application's error messages are compared for how many of these quality attributes are fulfilled. With RQ3 we also identify common mistakes that are made when writing error messages. We will also give solutions and ideas to these common mistakes.

The remainder of this thesis is structured in the following way: In Chapter 2 we show current scientific literature about specifications, error injections and error messages. Our approach and methodology can be seen in Chapter 3 where we present our own written tool *Lyrebird* in more detail. Chapter 4 presents the results of our thesis and discusses

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the result at the end of the chapter. In the final Chapter 5 we sum up our findings and conclude the thesis.

Throughout our thesis we use the following terminology:

- configuration: A file which is used by an application to configure its behavior (e.g, an `ini` configuration file)
- configuration setting: A concrete setting in a configuration (e.g., `server_address = localhost`)
- key or configuration setting key: The concrete key of a configuration setting (e.g, `server_address` in `server_address = localhost`)
- value or configuration setting value: The concrete value of a configuration setting (e.g, `localhost` in `server_address = localhost`)
- configuration solution: A concrete instance of a configuration with various configuration settings
- specification or configuration specification: Additional metadata which constrain and describe configuration settings
- specification language: The language to describe configuration setting as well as its constraints
- misconfiguration or invalid configuration setting (ICS): A concrete configuration setting which will cause the application to crash or function in unwanted ways. An example would be if a user writes `localhoost` as connection address for an application when he really meant `localhost`. We may also refer to the configuration of the system on which the application runs and interacts since file/directory permissions for example are also needed to be correct for many applications to work correctly (e.g., for writing log files)
- configuration error: A configuration solution which contains at least one ICS that causes the application to crash or function in unwanted ways
- medium-sized application: An application with approximately 150 - 300 configuration settings

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

CHAPTER 2

State of the Art

2.1 Configuration Specification

We will now present the state of the art papers and specification languages which have their focus in software configuration. As of today, we could not find any notable paper about specification languages with an emphasizes on configurations except of *SpecElektra*.

SpecElektra *SpecElektra* was developed by Raab[36] and is a specification language. SpecElektra allows us to specify configuration settings along with their constraints. This thesis also takes usage of SpecElektra as all specifications which are written use the specification language of it.

Windows Registry Apart from SpecElektra there are some configuration specifications built-in, into bigger software systems such as Microsoft's Registry. Each configuration setting in the registry must be associated with a specific type which then gets validated. Examples are checks if the configuration setting value is a 32-bit number or if the given value is an unexpanded reference to environment variables [28].

KConfig XT Another configuration specification language is built-in, into KDE applications which build upon KConfig XT, a configuration framework[22]. The configuration framework requires XML files in a certain structure and allow us to check for various types such as Strings, Fonts, Colors, Enums, Path, Url, etc. Also you can provide ranges for integers which then allows users to put in only these valid numbers in the user interface.

Zero-configuration It is also noteworthy to mention that some software applications completely mitigate the need for configurations by using a zero-configuration approach. A prominent example is GPSD [37], a suite of tools managing collections of GPS devices.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

They rely on auto-detection and autoconfiguration so that users do not need to specify any configuration settings.

2.2 Error Injection

Another aspect of our thesis are error injections into configurations. There has already been a lot of research in this area.

Conferr ICS injections in configurations are not new and already has been done by Keller, Upadhyaya, and Candea[6]. The pioneer in this area uses a model to inject ICS that are based on linguists and psychologists studies of human behavior. The tool automatically injects ICS in configurations by using structural errors, typo errors as well as semantic errors (which are only used for DNS entries).

Confvd The newest and more sophisticated approach was done by Li et al.[8]. The authors examined eight mature open source and commercial software applications and classified the configuration setting types. Since this paper is the most recent and modern approach we want to give a deeper insight into their approach.

Li et al. applied a fine-grained classification by extracting syntactic and semantic constraints for each configuration setting value in order to generate ICS. These ICS are injected into the applications and the reaction of the system is analyzed. For this process, Li et al. developed a tool called ConfVD which automates the task of injecting ICS.

Each configuration setting is given certain constraints which it must satisfy in order to be valid. Examples are file path configuration settings or boolean configuration settings. To find such constraints, Li et al. researched configuration settings from Squid, Nginx, Redis, Nagios, Lighttpd (core), Puppet, SeaFile, Vsftpd which are well-used in their field. A manual investigation has been carried out by using official documentation as well as the application's configuration to extract detailed information.

For a closer look of the configuration setting type classification the reader is encouraged to take a look into the paper of [8, p. 3]. Compared to other type taxonomies, the focus on ConfVD's classification lies in constraint generation for configuration settings which can be used for ICS injection.

The classification tree is applied to four software systems: Httpd, Yum, PostgreSQL and MySQL. It covers 96,5% of all types excluding "Others". The type "Others" is used for configuration settings that are hybrid such as the "LogFile" in MySQL which can either be a network address or a system path. The proposed classification is meant to be easily expansible with new types if there is the need to classify a particular type of configuration settings.

Fine-grained constraints were developed for each configuration setting. Domain knowledge as well as inherent constraints were used for this. Program analysis for constraints

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

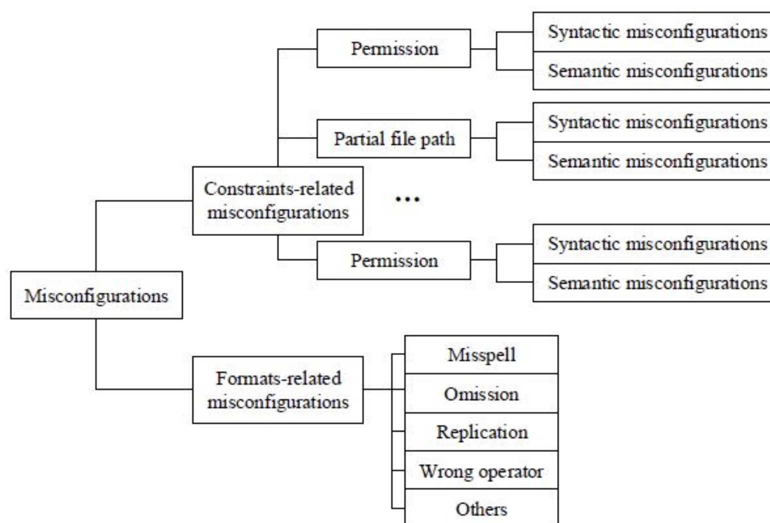

Figure 2.1: Misconfiguration generation rules[8]

generation was not executed because it is too difficult if not impossible. Compared to other papers, Li et al. considered syntactic as well as semantic aspects. One example is a PORT configuration setting which syntactically has to be an integer between 0 and 65535 and should not be in use by another application.

Syntactic constraints come in the form of string patterns like regular expressions. This approach was also done by EnCore [44]. An example would be a `SysadminName` which has to fulfill the following syntactic constraint `[a-zA-Z][a-zA-Z0-9]*` in order to be valid. Out of the 1582 configuration settings from Httpd, MySQL, PostgreSQL and Yum the proposed syntactic constraints are consistent by 91%. The reader is encouraged to take a look into the paper of Li et al.[8, p. 5] to get a deeper insight into those string patterns.

To generate misconfiguration, the tool ConfVD parses configuration files into a structured format and then modifies the data. This tampered configuration is then used by the target system. The rules which are used to classify injections can be seen in Figure 2.1. ConfVD uses constraint-related and format-related injections.

The violation of constraints in the configuration is done by constraint-related rules. For example in Httpd the `Listen` configuration setting is of type PORT. The syntactic constraints are

- higher or equal to 0
- lower or equal to 65535
- should be an integer

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Software	LoC	Options	Misconfigurations	Reactions (Duplicated ones removed)
Httpd	148K	29	328	78
MySQL	1.2M	26	318	113
PostgreSQL	757K	33	352	10
Yum	38K	25	275	15

Table 2.1: Analyzed Systems[8]

Abbr.	Yum	Httpd	PostgreSQL	MySQL	Sum	Ratio
Good Reaction	7	23	5	37	75	34.56%
Bad Reaction	3	11	3	48	64	29.95%
No Reaction	5	41	3	28	77	35.49%
Sum	15	78	11	113	217	100%

Table 2.2: System reactions results[8]

To violate a semantic constraint the injected port is actually used by another application.

ConfVD injects typical user mistakes such as misspelling, omission, etc. when editing configuration files. To test for correctness of the software, infrastructure test cases as well as test oracles are used by ConfVD. These test cases come from the application themselves such as the test suite for MySQL which is available online[29]. The behavior of an administrator are simulated and sequential steps are executed such as launching the system, functional tests, etc. The erroneous configuration with the ICS replace the standard one on the target system. If the target system successfully started, functional tests are executed and the system either fails or succeeds for all cases. All output and logs are analyzed in the process and afterwards evaluated for the system reaction analysis.

The 1273 injected misconfigurations can be seen in Table 2.1. Due explosive exponential growth of injection possibilities, only a sample of all possible misconfigurations is taken. Li et al. show that simple configuration settings are easier validated compared to path related configuration settings. The result of their classification can be seen in Figure 2.2.

Apart from the paper of [8], misconfiguration detection and troubleshooting has been subject in many other scientific papers. Also classifications of configuration settings has been dealt with such as the paper of Liao et al.[9] which find constraints of configuration setting via studying five commonly used open source software. Their classification aims to improve automatic configuration setting constraints extraction.

ConfSuggester & MisconfDoctor Some papers have a different approach of finding misconfiguration such as ConfSuggester [46]. This tool developed by Zhang and Ernst uses dynamic profiling, execution trace comparison and static analysis to link an undesired behavior to its root cause in case of a misconfiguration. This log-based configuration testing approach was also done by MisconfDoctor[40]. Log features are extracted for

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

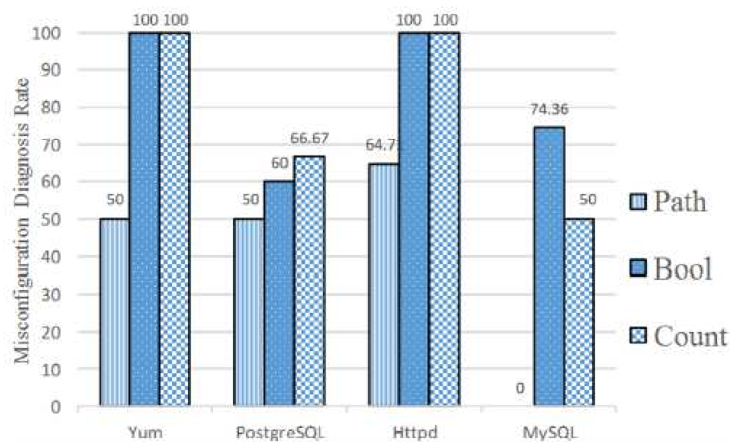

Figure 2.2: Misconfiguration diagnosis rate of different systems with three kinds of misconfiguration[8]

every misconfiguration and saved into a database. If another misconfiguration happened, the tool searches the database for similar cases by comparing logs and suggests a possible misconfiguration.

Decision Tree Analysis To find misconfiguration in cloud datacenters, Uchiumi, Kikuchi, and Matsumoto[39] developed a decision tree analysis for misconfiguration detection. It identifies a relation among the majority of parameters by doing a statistical decision tree analysis. By applying pattern modifications this method was able to detect 78,6% of misconfigurations.

ConfValley Huang et al.[5] created a declarative language to express configuration specification. The language is used to detect configuration errors in the latest Microsoft Azure deployments.

ConfDiagDetector Via configuration mutation and natural language processing the tool ConfDiagDetector tries to detect errors and diagnose them. Their focus is not on finding misbehavior but to detect inadequate messages [45].

2.3 Error Messages

When switching from configuration validation via code to a (possibly generic) validation via configuration specification, the error messages should at least be equally good in guiding users to fix the misconfiguration via an error message. The majority of papers deal with designing good error messages for novice programmers which can also be applied to misconfiguration messages.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Students interaction with DrRacket Marceau, Fisler, and Krishnamurthi for example investigate students reaction to DrRacket, a programming language which put considerable effort into the design of the error messages. They show that reduced vocabulary and message grammar improve the understanding of the error message. Also they show that users prefer an "edit here" mentality when it comes to highlighting important parts of the error message. Highlighting correct parts of a program would result in changes there, which would worsen matters with each iteration. They also found that consistency is of importance for new users as different notions for the same subject will lead to confusion. This is also the reason why they suggest to be careful when using libraries as they bring inconsistency [10].

Students interaction with Pyret For the teaching programming language Pyret, Wrenn and Krishnamurthi apply a precision and recall metric on error messages. They also show in their survey, that their implemented highlighting helped users with few exceptions. Some users complained about the verbose error message which shows the tension between providing enough information but not overwhelmingly much [42].

Personification Lee and Ko added a personal pronoun such as "I" or "you" to normal messages and admitting of failure such as "I don't know this command" in error messages. They show that personifying the computer and making it less authoritative gives learning benefits and improves motivation [7].

Apologizing Giving computer messages a more humanly touch was also researched by Park, MacDonald, and Khoo. They showed that an apologetic error message leads to an increase in perceived aesthetics and usability. However, when displaying such messages they have to be succinct, unique and noticeable to be authentic and non-exaggerated [34].

Security in Error Messages Brown draw attention to security in error messages since leaked internals could potentially give attackers important information [1].

Suggesting Solutions Where scientific literature is not in a complete agreement are solution suggestions. While [10] and [38] do advise against giving solution due to possible misleading in special circumstances, [3], [4] and [41] suggest to provide a solution.

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 3

Methodology

In order to show how time intensive the writing of specifications are, we write a specification for Apache Cassandra, a database tool as well as LCDproc, a display software for system information. Both applications have around 190-300 configuration settings. We also researched possible extensions for Elektra which can be used to improve the specification language. We tracked the time needed for writing the specifications of both applications as well as the time needed to develop three additional extensions for Elektra. With this data we will be able to answer RQ1.

The written extensions for Elektra will enhance the following fields:

- Permissions on path configuration settings: We enhanced the existing `path` extension with the capability of checking for correct permissions. As Li et al. state, it is comparably difficult to check for path related configuration settings[8].
- Checks for occupied ports: We enhanced the existing `network` extension with the capability to check for occupied ports. Especially in Cassandra's configuration there are many port-related configuration settings.
- Checks for configuration settings referring to other configuration settings: A colleague tracked the time needed for developing the `recursion` extension which checks for endless loops of configuration settings referring to each other. In a client of LCDproc, users can configure a dynamic menu in the configuration with menus referring to other menus.

To answer RQ2 we investigated how many ICS can be caught by the application. We then take the same ICS and see how many Elektra catches. There are different forms of ICS which administrators and users may do such as typo or semantic errors. We categorized ICS into six error types and give more detail about these error types in the next section.

We also wrote the tool *Lyrebird* that automatically can inject ICS of such error types into configurations and gather logs for further analysis. *Lyrebird* will be presented in more detail in Section 3.2. With *Lyrebird* we can then compare the ICS detection rate of the application with Elektra and analyze their respective performance.

3.1 Error Types

3.1.1 Structural Errors

Section Removal

Before	After

# Other configuration settings 	
server:
    port: 8080
    host: localhost
    protocol: TCP
menu:
    foreground: blue
# Other configuration settings 	
	# Other configuration settings
	# server removed
	menu:
	    foreground: blue
	# Other configuration settings

In the upper example the `server` configuration setting was removed along with all configuration settings that are below. A single configuration setting may also be removed rather than a whole section.

Section Reallocation

Before	After

# Other configuration settings 	
server:
    path: srv/driver/
    port:8080
menu:
    foreground: blue
# Other configuration settings 	
	# Other configuration settings
	server:
	    port: 8080
	menu:
	    foreground: blue
	    path: srv/driver/
	# Other configuration settings

In the upper example the `path` configuration setting was moved from the `server` section to the `menu` section.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Section Duplication

Before	After

# Other configuration settings 	
server:
    port:8080
menu:
    color: blue
# Other configuration settings 	
	# Other configuration settings
	server:
	    port: 8080
	    menu:
	        color: blue
	menu:
	    color: blue
	# Other configuration settings

In the upper example the whole `menu` section was copied below the `server` section. Some special configurations will also be injected which have the section at the same level like this:

Before	After

# Other configuration settings 	
menu:
    foreground: blue
# Other configuration settings 	
	# Other configuration settings
	menu:
	    foreground: blue
	menu:
	    foreground: blue
	# Other configuration settings

3.1.2 Semantic Errors

Semantic errors include all kinds of errors in which the developer of the configuration has a different understanding compared to the user/administrator who changes it to fit their special needs. Examples are wrong versions, similar applications, false enumeration options (eg. an invalid color option), misunderstood operators (eg. a valid display size declaration in the form `20x4` will be written as `20*4`), etc.

3.1.3 Resource Errors

Resource errors are all kind of permission and existence errors that are essential for the application to operate. Examples are missing files/directories or insufficient permission to execute certain commands (e.g., you would require sudo permissions) or read/write for certain files.

3.1.4 Typo Errors

Typo errors are all sorts of mistyped words. The following table shows possible variants for all kinds of typo errors for the word `foo bar`.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Error Type	Result
Transposition	`foo abr`
Insertion	`foo baGr`
Deletion	`foo ar`
Char Changing	`fow bar`
Space Typo	`'foo bar '`
Case Toggling	`foO bar`

3.1.5 Domain Errors

Domain errors are specific misunderstandings such as not an URL for a start page configuration setting or if an application requires a numeric IP but the user provides `localhost`.

3.1.6 Off-by-one Errors

Limit errors are edge cases where the upper and lower limit of range configuration settings will be taken. Off-by-one errors can easily cause the application to fail such as loops which exceed the maximum element size. An example would be a brightness configuration setting which allows a value between 0 and 1000 (both inclusive) where the injection will either take 0 or 1000 and start the application.

3.2 Automatic Invalid Configuration Setting Injection - Lyrebird

In order to make ICS injection in applications standardized and more reliable we wrote an application called *Lyrebird* which is freely available on GitHub [47]. *Lyrebird* is a Java based application which can show the performance of the ICS detection rate of an application as well as a specification for the application which was written in the specification language of Elektra. Figure 3.1 shows a high level view of *Lyrebird* testing an application.

The tool first loads the configuration from the application in memory and also a prepared injection configuration. The injection configuration is a copy of the underlying and working configuration from the application itself but with additional metadata that will be used by *Lyrebird* to inject an ICS. An injection configuration holds all possible ICS for each configuration setting as well as the default value which would be used by the application if the configuration setting would be absent. The default value is used by typo errors in case no value is given by the application's configuration (e.g., it is commented out because it should only be set when using a different value from the default). Typo and structure errors can always be applied, but domain-, semantic-, resource- and off-by-one errors will be loaded from the configuration setting if they are available. For typo and structure errors, *Lyrebird* pseudorandomly chooses which changes it applies, e.g., which char to change, where in the configuration is should duplicate a section, which random

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

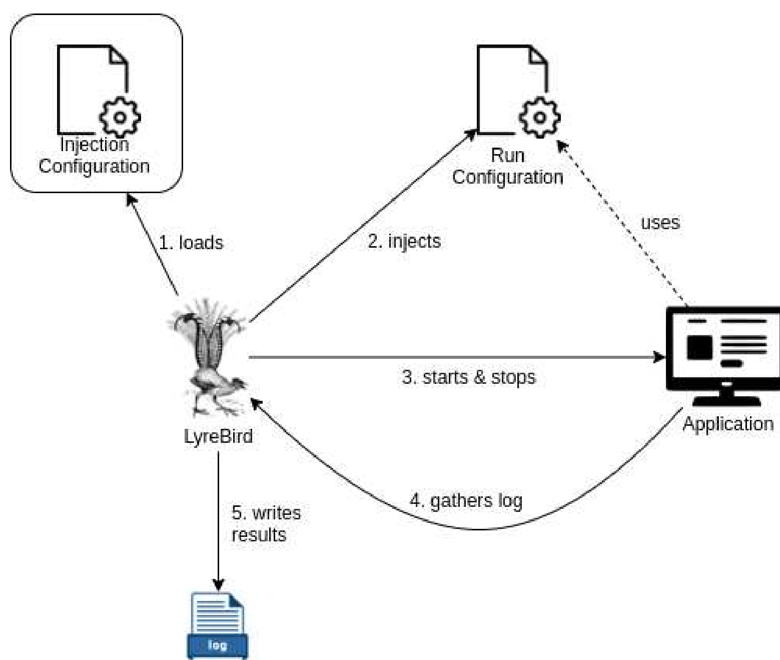

Figure 3.1: Top down view on how *Lyrebird* works with the application.

char it takes to substitute another char, etc. For the other error types we thought of possible ICS like different forms of boolean expressions (e.g., yes, on, 1, y instead of `true`), server ports which are actually in use already by the operating system, optic similar but invalid expressions such as 20*4 instead of 20x4 which is expected, different forms of color representations for e.g., a background, etc. and added it to applicable configuration settings. All concrete injections for each configuration setting can be seen in the Github repository of *Lyrebird* [47].

An example entry can be seen in Listing 3.1 which shows all injection possibilities for a configuration setting that expects an existing Linux device.

Listing 3.1: Example injection configuration entry which expects a path to a device

```
[/Device]
default = /dev/tty
inject/semantic/#0 = /usr/bin/gnome-terminal ;Similar but no text-
    ↪ only console
inject/domain/#0 = eth0  ;Misunderstanding of the Domain, actually an
    ↪  interface
inject/resource/#0 = /dev/not_existing ;Resource not available
inject/resource/#1 = /etc/shadow ;Resource available but insufficient
    ↪  permission
```

Since the `Device` configuration setting is no number with upper or lower bounds, the injection configuration does not contain an off-by-one error for this specific configuration

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

setting.

Once the injection configuration is loaded, among all available configuration settings a single one is pseudorandomly chosen. In the next step *Lyrebird* loads all metadata of the chosen configuration setting which holds the possible injection strategies. *Lyrebird* randomly takes one of the possible injection strategies and applies it to the actual configuration from the application. For every run, only a single injection takes place and is not combined (e.g, applying a domain error and additionally a typo error).

Lyrebird can also change additional configuration settings which forces the injected configuration setting to be used by the application. For example *Lyrebird* creates an empty file if it detects that a certain configuration setting expects it to be present under a certain path before validating any subsequent configuration setting. For this we manually analyzed configuration settings beforehand and specifically implemented this behavior for configuration settings of LCDproc and Cassandra.

In some cases we even manually executed injections when the relevant configuration settings needed specific other complex configuration settings enabled. For example, to check for an ICS of a SSL port configuration setting in Cassandra, another configuration setting must be set to `true` which tells Cassandra that encryption should be used and subsequently the SSL port.

Proceeding the Example 3.1, *Lyrebird* randomly can choose among the following options:

- Typo Error
  - Transposition: e.g., /dev/tyt
  - Insertion: e.g., /devT/ttY
  - Deletion: e.g., /dev/tt
  - Char Changing: e.g., /dev/ttx
  - Space Typo: e.g., ' /dev/tty'
  - Case Toggling: e.g., /dev/tTy
- Structure Error
  - Section removal: Deleting the entry
  - Section reallocation: e.g., /Device moved to /server/Device
  - Section duplication: e.g., /Device copied to /server/Device
- Semantic Error
  - /usr/bin/gnome-terminal
- Domain Error
  - eth0

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

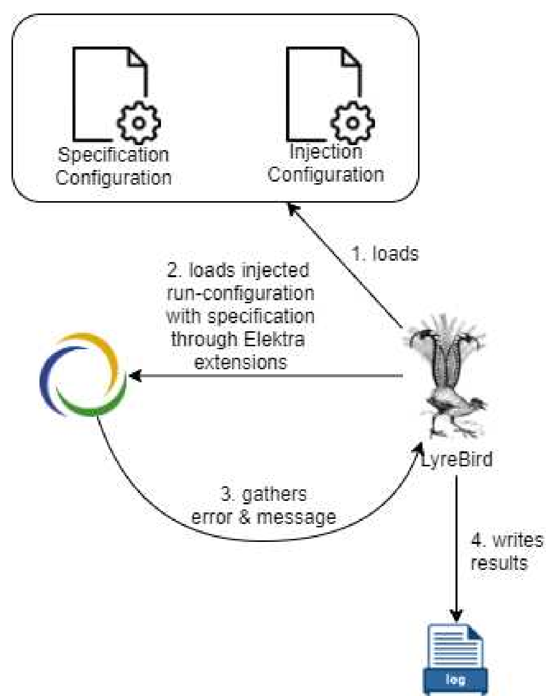

Figure 3.2: Top down view on how *Lyrebird* works with the Elektra.

- Resource Error
  - /dev/not_existing
  - /etc/shadow

Every randomness which is supplied by *Lyrebird* is pseudo randomness to guarantee reproducibility. *Lyrebird* starts the application with the erroneous configuration and saves all logs which are generated in the process. After a predefined time for the application to start up, *Lyrebird* stops it again. The waiting time can vary depending on the application (e.g., databases usually take longer to startup than embedded command line based applications) but is hard coded for the application and based on the standard time which is used on the authors system.

The same injection procedure is also done with Elektra and depicted in Figure 3.2. The same ICS are loaded through several extensions of Elektra which validate configuration settings based on a specification. If Elektra catches the ICS, the error message is saved.

The log messages of the application as well as the error message from Elektra is then written into a file for further manual analysis for the quality of the reactions.

We use four types of reactions which are similar to [8]Li et al. to better analyze reactions of Elektra and the application:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- *Good reactions*: Misconfiguration is explicitly located. The reaction is directly related to the injected error but the context/ configuration setting location must not be given explicitly since it will be analyzed in the following steps for the quality. Good reactions happen during system startup when the configuration is checked for validity or in case of Elektra, once the configuration setting is changed/created.
- *Bad reactions*: An exception is triggered, but the concrete reaction of the system obscures or misleads the diagnosis of the actual problem.
- *No reactions*: No exception related to the misconfiguration is found during startup. This means either latent errors or unused configuration settings.
- *No reaction needed/Failsafe mechanism triggered*: Exceptions are not expected to occur if the application can correctly handle the configuration setting such as off-by-one errors in which the upper and lower bounds are correctly implemented. Another example would be typo errors in a welcome message or any display text which does not cause the application to behave in unintended ways. No reactions can also occur on configuration settings which were removed either by reallocation of the configuration setting or complete deletion. The application often takes a default value in this case. Applications can also have a failsafe mechanism implemented such as multiple forms of boolean definitions which help to mitigate semantic errors of this kind.

All saved log messages from the application as well as error messages from Elektra are then subject to further analysis. All *Good reactions* will be categorized whether they fulfill quality attributes of good error messages. These quality attributes have their roots in scientific literature and consist of:

1. Fully pinpoints the ICS. This attribute is completely fulfilled if the error message states a) the configuration key, b) the configuration value, c) a reasonable context to solve the ICS

   E.g., if the user typed in an illegal enum, the application should give information about the configuration key, the current value and valid alternatives

2. Applied meaningful highlighting to guide users

   Highlight the concrete error or error code

3. Use of reduced and consistent vocabulary

   E.g., do not use special abbreviations or words which require advanced knowledge in the topic

4. Personified and apologetic error message [34][7]

   E.g., *Sorry, I was not able to parse XY* instead of *XY has invalid syntax*

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5. Does not leak internals

   E.g., a bad error message would include method names and parameters, stack traces, variable names in the error message, etc.

We decided to not include the proposal of a solution in the error message since it is controversial in scientific literature. Which attributes are fulfilled by either Elektra and the application will then enable us to answer RQ3.

Lyrebird is conceptually similar to ConfVD [8] and ConfErr [6] but differs in its purpose and execution. While ConfErr and ConfVD's primarily purpose is to detect bad system reactions, *Lyrebird* has its emphasize in comparing the applications reaction to ICS with the configuration framework Elektra and see if a specification can enhance the system's reaction. *Lyrebird* also uses different kinds of error types. ConfVD also tests a subset of configuration settings very thoroughly (e.g., all available error types for a configuration setting) whereas *Lyrebird* tries to inject ICS in many different configuration settings but not every available error type possible.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Evaluation

4.1 Evaluated Applications

This section gives a more detailed picture of the investigated applications.

4.1.1 LCDproc

On the welcome page of LCDproc a good description is given of the software's purpose:

> LCDproc is a piece of open source software that displays real-time system information from your Linux/*BSD box on a LCD. The server supports several serial and USB devices from Matrix Orbital and CrystalFontz as well as some devices connected to the LPT port: HD44780, T6963, SED1520 and SED1330. Various clients are available that display things like CPU load, system load, memory usage, uptime, and a lot more [24].

As of version 0.5.7, LCDproc features 330 configuration settings for the server. The project itself has about 27 contributors, ~135,000 lines of code and ~2,200 commits as of 07-10-2019. The project is written in C and the authors of LCDproc have written their own manual configuration parser and did not use a library parser which could potentially have more sophisticated checks on the structure of the configuration files. This comes most likely due to the limited space of computers where LCDproc is usually installed.

LCDproc features 52 display drivers where on our system we were not able to install 11 of them via the package manager and the additional lcdproc-extra-drivers package. This leaves 295 remaining configuration settings to test.

Table 4.1 shows the statistics of LCDproc's configuration settings. The majority of the configuration settings are strings. That's because many configuration settings configure

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Type	Occurrence Count	Percent
boolean	55	18.64%
enum	22	7.46%
float	1	0.34%
string	142	48.14%
unsigned_long	10	3.39%
unsigned_short	65	22.03%
Sum	295	100.00%

Table 4.1: Server configuration setting type statistics of LCDproc v0.5.7 for 41 drivers

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

key bindings such as `Enter` or arrow keys to navigate through the menu with multiple drivers. Additionally, LCDproc compacted configuration settings such as display sizes. Rather than providing separate height and width in pixels for the display device dimensions they have a single configuration setting in the form of "[Number]x[Number]". Another noticeable fact are the various inconsistencies over all configuration settings. One example are multiple boolean definitions such as true/false, on/off, yes/no. Also defined ranges are not consistent as for example most drivers set the display brightness between 0-1000 while some range from 1-1000. Some drivers are also unmaintained and outdated. Additionally it is worth noting that some configuration settings are repeated for many drivers such as the `Brightness` which occurs for 12 drivers or the size declaration which occurs 22 times.

4.1.2 Cassandra

Apache Cassandra is an actively developed database software which emphasizes on scalability and high availability without performance drawbacks. Partitioning is also supported which enables distribution across multiple machines. As of 07-10-2019, the Github website[13] shows 271 contributors, ~710,000 lines of code and ~25,600 commits. While being much bigger in terms of code base, popularity and contributors compared to LCDproc, Cassandra has just about ~190 configuration settings. The configuration settings count can vary because some configuration settings depend on the number of cluster nodes (e.g., the user has to provide additional IP addresses).

Table 4.2 shows all configuration settings for Cassandra. Cassandras configuration settings are primarily numeric configuration settings. Strings are predominantly class names which have to be provided (including class parameters) in case a user needs some special behavior. Apart from that, there are also 10 path configuration settings such as log location, hints directory, data directory, etc. which are categorized as strings.

Compared to LCDproc, Cassandra has a very standardized and consistent configuration with detailed documentation. Configuration settings in general are easier to understand compared to LCDproc as the configuration setting keys are more verbose and self-explanatory.

Type	Occurrence Count	Percent
unsigned_long	83	44.62%
string	49	26.34%
boolean	25	13.44%
enum	14	7.53%
unsigned_short	5	2.69%
long	4	2.15%
float	4	2.15%
unsigned_int	2	1.08%
Sum	186	100.00%

Table 4.2: Configuration setting types statistics of Cassandra v3.11.4

4.1.3 Elektra

Elektra is an open source modular configuration framework. Compared to Cassandra and LCDproc, Elektra's main purpose in our thesis is the use of its configuration specification language which is integrated in the module `SpecElektra`. With Elektra we can overlay a configuration specification over the application's configuration. If we change a configuration setting which violates the specification, Elektra will emit an error and roll back any changes so the configuration is in a legal state. In our thesis we will compare this violation detection from Elektra's specification module to the native application's configuration validation. Our self written tool *Lyrebird* will execute an ICS in both Elektra and the compared application to analyze the system reaction.

We initially made our injection runs with Elektra v0.9 where we overhauled the complete error code system ourselves and enhanced/ fixed many of the 660 error messages. We also received help from the maintainers of Elektra in these regards as they have the deepest insight into the system. Also many code sections were unclear which had influence on the quality of the error messages where we had to change them. We also consulted help from the original code writers in these regards if we were not able to understand the error correctly. After our injection runs we have seen some flaws in the error messages which resulted in additional changes which can be seen in version v0.9.1. These enhancements were a direct result of our thesis [16][15].

4.2 LCDproc Results

Only 3 drivers are actually supported by a standard Linux Laptop so in most cases the application will not start which does not affect results of the systems reaction. It is assured that every changed configuration setting is loaded by the application. Furthermore every driver expects a `Device` configuration setting which depends on the display being used. *Lyrebird* creates an empty file for these drivers so it can still test the systems reactions as the code which parses the configuration only checks for the existence of the file. If the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Misconfiguration Error	Occurrence Count	Percent
Section Duplication Error	59	12.4%
Section Reallocation Error	63	13.3%
Section Removal Error	59	12.4%
Semantic Error	59	12.4%
Resource Error	10	2.1%
Typo Char Transition Error	30	6.3%
Typo Char Insertion Error	36	7.6%
Typo Char Deletion Error	43	9.1%
Typo Char Changing Error	22	4.6%
Typo Space Insertion Error	25	5.3%
Typo Case Toggle Error	22	4.6%
Domain Error	20	4.2%
Max Off-by-one Error	13	2.7%
Min Off-by-one Error	13	2.7%
Sum	474	100%

Table 4.3: LCDproc configuration injections per concrete error type

application then throws an error which is related to the lack of correct hardware (e.g., some application listening on specific ports) and not the injected error, it is considered as *No reaction* because it results in a latent error since the configuration is already successfully loaded (and validated).

Table 4.3 shows the pseudorandom run of 500 injections. Duplicates have been removed which results in 474 unique ICS. The overall statistics are balanced when taking the underlying configuration settings into account (e.g., few off-by-one errors because of few range configuration settings or configuration settings which point to a directory/file for resource errors). Also domain errors are considerably few because it is only applied to senseful configuration settings such as color configuration settings which were given different forms of representations (e.g., `red` was changed to `#FF0000`) or if the configuration setting itself sounded as if it could expect a boolean but actually wanting something else (e.g., `normal_font` for the glcd driver expects a path to a .ttf file but could also be mistaken for turning on and off a special font).

4.2.1 Specification Evaluation

Writing the specification for the server configuration of LCDproc along with its three clients took the authors approximately 24 hours and 25 minutes. The reason for this comparably high time value comes from various reasons:

- Scattered and lack of documentation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

	Good	None	Bad	No reaction needed
Elektra	161	86	12	215
LCDproc	74	149	20	231

Table 4.4: System reactions of Elektra vs. LCDproc. Elektra catches errors via a self written specification whereas the application catches errors via a configuration validation which is integrated into the application itself

In the initial configuration of LCDproc every configuration setting is well commented concerning default values or valid enumerations. Unfortunately though some configuration settings are not very well documented on which values they can take such as `Key` configuration settings. The default configuration setting for giving the next screen in LCDproc for example is the `Right` key but is `Alt+F` also valid? Is upper or lower case relevant? Could I also use a tabulator as key and how? Is there a maximum value for timeout configuration settings? Can configuration settings with path' also be relative? Some of these questions could be answered by looking through LCDprocs .docbook files in their repository but many configuration settings required submitting questions to the developers of the drivers.

- Getting used to Elektra

  Getting used to the concept of mounting/namespaces/all available extensions/etc. took some time. Elektra uses many extensions with decent documentation for specification validation. Some extensions though were not working as expected and contained bugs [21][18][12][32][19][17][25][11]. Others were unclear in their usage or did not have appropriate documentation yet [31][30][20].

- Complex settings

  The specification of LCDproc has configuration settings which require a special handling such as a size declaration in the form of e.g., `20x3` or special port declarations with ranges such as `0x200 - 0x400` which took some longer tinkering with regular expression.

The majority of the time consuming tasks can be allocated to the lack of documentation. We noticed that an editor with multicursor support and good regular expression usage cut the time significantly as we had to edit many configuration settings in parallel quite often. The current specification file of LCDproc has 2700 lines and many configuration settings are very similar. A regular expression search is needed instead of a standard search since many configuration settings have their name also appear in the description somewhere which would cause also edits there which are unwanted.

Table 4.4 shows the comparison of reactions for the 474 misconfiguration runs. Table 4.6 gives a more detailed view on the reactions for each error type for LCDproc. The same evaluation can be seen for Elektra in Table 4.5.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

	Good	None	Bad	No reaction needed
Typo Error	115	54	0	9
Domain Error	7	1	12	0
Structure Error	0	1	0	180
Semantic Error	29	30	0	0
Off-by-one Error	0	0	0	26
Resource Error	10	0	0	0

Table 4.5: Reactions of Elektra per Error Type

	Good	None	Bad	No reaction needed
Typo Error	42	103	3	30
Domain Error	2	7	11	0
Structure Error	0	1	4	176
Semantic Error	22	37	0	0
Off-by-one Error	0	0	0	26
Resource Error	8	1	1	0

Table 4.6: Reactions of LCDproc per Error Type

Elektra reacted better in more than double the cases compared to LCDproc's native parsing. LCDproc mainly reacted very well on invalid size declarations which were available in every driver which made up the majority of the good reactions. It also reacted well on path related errors such as not enough permissions for opening a device file or wrong driver file path. Elektra in comparison was also able to catch invalid ranges (e.g., the `ReportLevel` configuration setting accepts values between 0-5 but threw no error when providing 72), invalid port ranges (e.g., `03x78` despite it expected a range between `0x200 - 0x400`), wrong color declarations (e.g., `reed` instead of `red`) and much more. It is also notable that 13 of LCDproc's reactions were categorized as `No reaction needed` which were categorized as `good` from Elektra since the parser of LCDproc ignores whitespaces and Elektra does not. In only 11 cases LCDproc was able to react to injected misconfigurations while Elektra could not. The majority of these errors were related to additional checks with the hardware such as keybindings or server IP addresses that are not possible for binding. Some misconfigurations were also not feasible to check against for Elektra such as a configuration setting for the lirc driver which expected a value to be equal to a configuration setting in another configuration file. For these kind of errors no specification could be written expect for making a very specific Elektra extension. We also found one interesting behavior when deleting the `Type` configuration setting for the sed1330 driver. LCDproc always takes a default value if the configuration setting is missing except for this one. It correctly writes an error message which Elektra does not since Elektra always provides a default value and can handle missing configuration settings.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

LCDproc did not react in 149 cases. Examples are changing the `OffBrightness` from 1000 to `g1000` in the CFontz driver where we did not receive any error. This resulted in some interesting behaviors such as changing the `Backlight` color from `red` to `rEd`. The background of the terminal window took the same color as the foreground color which resulted in unreadable text because they blend into each other. Also the `hd44780` driver did not react at all despite we double checked it in the code if the configuration is parsed before any hardware is accessed. This is most likely a bug in the driver. The hd44780 reactions made up to 24 of the `None` reactions from LCDproc. Elektra did not react to injections on configuration settings which had no specification written for as there was no appropriate validation extension available. For example the specification language Elektra cannot provide a way to check for keybindings since users can code their own specific ones which get validated and used on runtime by LCDproc itself. Also Elektra did not react to 19 injections which had different forms of boolean value representations such as changing `yes` to `on`, `1`, `Yes`, `true`, etc. This is because Elektra normalizes boolean values in the background and accepts a multitude of boolean representations. The accepting of a multitude of boolean value representations can also be seen in the yaml parsing of Cassandra. We also found a case where neither Elektra nor LCDproc resulted correctly on a section duplication error. When two `Driver` configuration settings were given, LCDproc tried to load both which resulted in unwanted behavior when trying to use the `text` and `curses` driver for example. Elektra always took the first occurrence in the configuration and ignored the second one.

Bad reactions are few for both Elektra and LCDproc. Both gave bad messages when it comes to domain errors. When providing a `Device` configuration setting, the domain error injected a network device such as `eth0`. This could be any other device descriptor too which the user might have misunderstood but the error message just showed that the file or directory is not available. This would let the user provide a full path to a potentially wrong device. LCDproc also gave wrong error messages for the serialVFD driver because it does not recognize the `Custom-Characters` configuration setting and will give an error message concerning this configuration setting on every injection which is misleading. Additionally we noticed a parser bug when duplicating a section in LCDproc which resulted in more bad behaviors. The original and working configuration looked as following:

```
[vlsys_m428]
SomeSettingA = valueA
Device = /valid/driver/path
SomeSettingB = valueB
```

After *Lyrebird* duplicates the `Device` configuration setting to the root content level as shown below (which can easily happen by users via copy-paste), Elektra restructures the file:

```
[vlsys_m428]
SomeSettingA = valueA
[]
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
Device = /valid/driver/path
vlsys_m428/Device = /valid/driver/path
vlsys_m428/SomeSettingB = valueB
```

The file is still valid but the self written parser is not capable of detecting this restructuring and takes the default value which might not work on the target system because the user intentionally changed it. The given error message is misleading for these cases: *vlsys_m428: could not open /dev/ttyUSB0 (No such file or directory)*. The user does not see his configuration setting getting used by the driver and probably gets on the wrong trail for a solution. We also found misleading cases in LCDproc where the error message converted a non existing path to a number which would confuse users and administrators: *lcdm001: open(37353984) failed (No such file or directory)*

`No reaction needed` came primarily from structure errors which made up about 176 reactions. The reason for this is that LCDproc takes the default value when it cannot find the respective configuration setting and does not throw an error. This could though result in confusion for relocation errors as the user assumes his configuration setting does not work. Also most duplication errors resulted in taking default values because of the bug shown above rather than the desired one without giving a message. The remaining errors are off-by-one misconfigurations which are handled correctly as well as ICS with whitespaces which are filtered by LCDproc. Also typo deletions from 1000 to 100 for example are valid configuration setting values and are accepted correctly. The whitespace errors are the reason why LCDproc has a higher number of no reactions expected.

4.2.2 Error Messages

Error messages are the essential part of finding the cause of a misbehavior and should therefore be of good quality. This section investigates all `Good reaction` messages which were issued by Elektra and LCDproc for the quality attributes given in Section 3.2. Since Elektra and LCDproc have a different count of `Good reactions` we additionally show their relative occurrence in % for better comparability.

Pinpointing

Giving the correct configuration setting is an essential part of a good error message. Also giving additional context such as allowed enumerations is important for users. If a message contains all three parts (key, value, reason with context) we classified the message as `Good`. If at least two but not all parts were provided we classified the message as `Partly`. If nothing at all or only one part was given we classified it as `Bad`. Such reactions can happen if an exception is thrown by the application which relates to the ICS but no further information is provided.

Table 4.7 shows the results of the error message quality of LCDproc in comparison to Elektra. Elektra performed better compared to LCDproc as more than half of the messages where `Good` whereas LCDproc reacted `Good` in one quarter of all cases. Especially path

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

	Good	%	Partly	%	Bad	%
Elektra v0.9.0	86	53.42%	75	46.58%	0	0.00%
LCDproc	18	24.00%	55	74.32%	1	1.35%
Elektra v0.9.1	161	100.00%	0	0.00%	0	0.00%

Table 4.7: Error message quality of Elektra vs. LCDproc

related errors, wrong enumerations and invalid types gave good messages such as: *The type 'enum' failed to match for 'system/lcdproc/curses/foreground' with string: 'rEd'. Allowed values: 'red' 'black' 'green' 'yellow' 'blue' 'magenta' 'cyan' 'white'.*

LCDproc sometimes managed to give an even more exact location of the ICS by providing the concrete line number: *Invalid character '/' on line 750 of /etc/lcdproc/LCDd.conf.*

LCDproc as well as Elektra often provided the invalid configuration value but not the concrete key on which the configuration setting is invalid. An example is an invalid `Speed` configuration setting on which LCDproc wrote *bayrad: illegal Speed 1201; must be one of 1200, 2400, 9600 or 19200; using default 9600* whereas Elektra wrote *Value '1201' not within range 1200, 2400, 9600, 19200.* Both did not provide the exact location but LCDproc at least gave the section to search in.

Elektra always provided at least the value and reason which resulted in no `Bad` reactions. The single bad reaction of LCDproc looks as following: *xosd: xosd_set_font() failed.* It resulted from a case toggling of the default font configuration setting.

LCDproc had some inconsistencies when it comes to error messages. A tampering of the `Size` configuration setting in different drivers for example resulted in error messages such as:

1. *ula200: cannot read Size s20x4* (typo insertion)
2. *curses: cannot read Size: 204; using default 20x4* (typo deletion)
3. *Invalid character 'x' on line 167 of /tmp/lcdd-tmp.conf* (semantic error which inserts whitespaces. 20x4 -> 20 x 4)

These messages from LCDproc resulted in different categorizations such as the last message being the most accurate one and being categorized as `Good`.

Highlighting

Meaningful highlighting was not applied for messages from LCDproc at all. Elektra on the other hand has a standard error message printing format which applies highlighting. Elektra fulfilled this requirement since the standard error message looks like the following:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

	Good	%	Bad	%
Elektra v0.9.0	124	77.02%	37	22.98%
LCDproc	72	97.30%	2	2.70%
Elektra v0.9.1	161	100.00%	0	0.00%

Table 4.8: Error message vocabulary of Elektra vs. LCDproc

```
Sorry, module `MODULE` issued error `NR`:
`ERROR_NR_DESCRIPTION`:  Validation of key "<key>" with string
"<value>" failed.
```

Consistent and Reduced Vocabulary

Table 4.8 shows that LCDproc performed better concerning vocabulary. The main reason comes from two error messages of Elektra that are commonly used which use the word `stat` when it cannot access a file or directory and `normalize` when Elektra tries to parse an invalid boolean value. Even though there is no standardized vocabulary for good error messages, we agreed that these two are too technical. LCDproc used the notion of a `shared object file` when it cannot find a given driver as well as the message *cannot sock_create_inet_socket: cannot bind to port...* which contains an internal method name and the notion of `port binding` which is too advanced for non-programmers we think. As a result we rated two messages as `Bad` concerning reduced vocabulary.

Overall though, both Elektra and LCDproc use consistent and reduced vocabulary. Especially Elektra is very consistent when it comes to presenting each message in the same format such as giving apostrophes around invalid configuration settings or giving the concrete error context in a *Reason:...* format.

Personification and Apologizing

LCDproc did not have any personified or apologetic error message as this form of message is quite uncommon. The standard error message from Elektra which can be seen in 4.2.2 supports such format and is applied to every error message because it uses a standard output format. As a result Elektra fulfilled this requirement.

We have seen parts in the source code of LCDproc which applies personification when the application is not run as super user. The stv5730 driver gives '*...: cannot get IO-permission for ...! Are we running as root?*' as error message when trying to access a port only allowed for the super user.

Internal Leakage

Table 4.9 shows the comparison of internal leakage from Elektra compared to LCDproc. Elektra did not leak any internas of LCDproc because it does not have access to it. LCDproc also performed well with the exception of two messages which pointed to a

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

	Good	%	Bad	%
Elektra	161	100%	0	0%
LCDproc	72	97.30%	2	2.70%

Table 4.9: Error message internal leakage of Elektra vs. LCDproc

Misconfiguration Error	Occurrence Count	Percent
Section Duplication Error	28	10.5%
Section Reallocation Error	25	9.4%
Section Removal Error	18	6.8%
Semantic Error	52	19.5%
Resource Error	33	12.4%
Typo Char Transition Error	13	4.9%
Typo Char Insertion Error	10	3.8%
Typo Char Deletion Error	14	5.3%
Typo Char Changing Error	14	5.3%
Typo Space Insertion Error	9	3.4%
Typo Case Toggle Error	6	2.3%
Domain Error	44	16.5%
Max Off-by-one Error	0	0.0%
Min Off-by-one Error	0	0.0%
Sum	266	100.0%

Table 4.10: Cassandra configuration injections per concrete error type

concrete method name in the source code which failed (e.g., *xosd_set_font() failed*). As Brown[1] pointed out in their paper it should be of no concern for the user of the internal workings. As a result we categorized these messages as `bad`.

4.3 Cassandra Results

The results of the injection runs can be seen in Table 4.10. We decided to only run 300 injections because the number of configuration settings is less compared to LCDproc. After duplicate removals we were left with 266 unique ICS injections. Despite having primarily numbers as configurations values there were only two possibilities (`otc_coalescing_enough_coalesced_messages` as well as `max_value_size_in_mb`) in which we had a clear documentation on possible off-by-one errors. The run though did not randomly choose these configuration settings with this error type. We gave every configuration setting which ends with `*_in_mb` or `*_in_ms` the possible domain error of being treated as boolean values (e.g., the user might misunderstand that a certain value he provides in another configuration setting should be treated as megabyte/milliseconds/etc.). The injection configuration contained

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

many semantic as well as domain errors because of similar configuration settings where they are well applicable. We gave number configuration settings which expect a positive number a possibility for a negative one (e.g., the user wants an "unlimited" timeout setting and assumes that -1 or 0 fulfills this requirement). Resource errors are also frequent because we provided a number that is actually bigger than an `unsigned_long` for many configuration settings. We thought of users might want have no limits in case of storage room or throughput throttling and provide a number that high so it fulfills this requirement similar to -1 or 0. Additionally there are many path related configuration settings which have resource misconfiguration possibilities such as an invalid path or a path with not enough permissions to read.

4.3.1 Specification Evaluation

The writing of 186 configuration settings took us 8 hours and 10 minutes. There are various reasons that this time is only about a third of the time from LCDproc.

- Less configuration settings: Cassandra has fewer configuration settings compared to LCDproc's server configuration as well as its three clients.
- Simpler configuration settings: Since the most difficult validations are string configuration settings (path', regular expressions, etc.) and Cassandra primarily has number configuration settings we were significantly faster.
- Less possibilities for validation: For string configuration settings we were not able to write a validation for 34 configuration settings. The specification language of Elektra cannot validate if a class is existent in a jar file and implements the correct interface. Some settings were also relative path configuration settings which are not yet possible in Elektra to validate against.
- Better documentation: Compared to LCDproc, Cassandra gave a much more detailed description of their configuration settings. Not only was the documentation in the configuration much more verbose, but open questions could be researched onto their homepage or stackoverflow. There was no need for a direct communication with the maintainers.
- Gained expertise in Elektra: Since the specification for Cassandra was our second specification we already had much more experience with Elektra. Also some bugs were fixed until we reached a point were we would ran into them (or at least knew of them before).

Table 4.11 shows the overall performance of Elektra compared to Cassandra. A more detailed look per error type can be seen in Table 4.12 for Cassandra and Table 4.13 for Elektra.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

	Good	None	Bad	No reaction needed
Elektra	135	42	0	89
Cassandra	159	26	13	68

Table 4.11: Reactions of Cassandra vs. LCDproc

	Good	None	Bad	No reaction needed
Typo Error	51	6	2	7
Domain Error	44	0	0	0
Structure Error	8	0	7	56
Semantic Error	24	19	4	5
Limit Error	0	0	0	0
Resource Error	32	1	0	0

Table 4.12: Reactions of Cassandra per Error Type

	Good	None	Bad	No reaction needed
Typo Error	33	26	0	7
Domain Error	41	3	0	0
Structure Error	0	0	0	71
Semantic Error	28	19	0	5
Limit Error	0	0	0	0
Resource Error	33	0	0	0

Table 4.13: Reactions of Elektra per Error Type

Cassandra in general reacted better in terms of `Good` and `None` reactions. In 34 cases, Cassandra was able to detect a misconfiguration and react better compared to Elektra. The reason for this comes from complex configuration settings which do not have a possible validation in the specification language of Elektra. The majority of these are configuration settings which expect a Java class that implements certain methods. Wrong relative path' configuration setting, invalid keystore passwords, IP addresses which are valid but unavailable for binding, illegal cipher suites, etc. were also among those cases. These cases also make up all the `None` reactions from Elektra. Elektra was able to catch 19 misconfigurations which Cassandra were not detecting. The majority of these were negative configuration values of configuration settings which have to be positive such as timeout-, interval-, threshold configuration settings. Also a case of a latent error was detected when providing a directory to `cdc_raw_directory` on which Cassandra has no permissions to. Typos such as `020` instead of `200` were also not caught by Cassandra. Theses misconfigurations also make up the `None` reactions of Cassandra. In only 5 cases neither Cassandra nor Elektra were able to react. Cassandra was also able to catch reallocation errors by giving the following message for example: *Invalid yaml. Please remove properties [enable_user_defined_functions] from your cassandra.yaml.* It though

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek, WIEN Your knowledge hub

	Good	%	Partly	%	Bad	%
Elektra v0.9.0	124	91.85%	11	8.15%	0	0.00%
Cassandra	104	65.41%	44	27.67%	11	6.92%
Elektra v0.9.1	135	100.00%	0	0.00%	0	0.00%

Table 4.14: Error message quality of Elektra vs. Cassandra

was only able to catch the minority of such errors and in case of reallocation errors of configuration settings for Java class parameters it even failed to start up (e.g., for the configuration setting `back_pressure_strategy/class_name`).

The 13 ICS which had bad reactions were primarily structure errors which yielded `NoSuchMethodException` from parameterized classes which lead to configuration settings that are actually not involved. Some IllegalArgument exceptions as well as Nullpointer exceptions with an error message cause of `Null` were also categorized as bad reaction because they do not help in finding the root cause.

The majority of no reactions needed come from structure errors which both Cassandra as well as Elektra can handle with default values or ignoring of duplicated sections. Cassandra was also able to handle a multitude of boolean expressions such as `on`, `yes`, `true`, etc. which resulted in the 5 semantic errors which were handled well. The typo errors which were handled correctly are errors which again lead to a valid number (e.g., typo deletion 100 -> 10).

Pinpointing

Table 4.14 shows the error message quality of Elektra vs. Cassandra in terms of pinpointing the problem.

Elektra performed considerably better when it comes to providing the exact location of the ICS. Cassandra had difficulties when trying to provide the value which the user typed in. As an example when changing the boolean configuration value `false` to `yalse` on the `cdc_enabled` configuration key the error message (without providing the full stack trace) looked as following:

```
ERROR [main] 2019-09-06 10:41:13,896 CassandraDaemon.java:749 -
    Exception encountered during startup: Invalid yaml: file:/
   home/wespe/.ccm/LyreBirdCluster/node1/conf/cassandra.yaml
Error: null; Can't construct a java object for tag:yaml.org
   ,2002:org.apache.cassandra.config.Config; exception=Cannot
   create property=cdc_enabled for JavaBean=org.apache.
   cassandra.config.Config@4f51b3e0; Can not set boolean field
org.apache.cassandra.config.Config.cdc_enabled to null value;
   in 'reader', line 1, column 1:
       authenticator: AllowAllAuthenticator
       ^
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

It seems that Java cannot provide the given value if it is not of the correct type and returns null instead.

The 11 bad pinpointing error messages from Cassandra relate to messages which give well enough context but neither they provided value nor the key. For example when providing -1 to the `counter_write_request_timeout_in_ms` configuration setting the error message yields:

```
ERROR [ScheduledTasks:1] 2019-09-06 10:37:32,139
    DebuggableThreadPoolExecutor.java:239 - Error in
    ThreadPoolExecutor
java.lang.ExceptionInInitializerError: null
Caused by: java.lang.IllegalArgumentException: Argument
    specified must be a positive number
```

Elektra on the other hand gives the following message in v0.9.1:

```
Sorry, module type issued the error C03200:
Validation Semantic: The type 'unsigned_long' failed to match
    for 'user/cassandra/counter_write_request_timeout_in_ms'
    with string '-1'
```

which provides all necessary information. In Elektra v0.9.1 the 11 partly good reacting messages were fixed. They did not provide the concrete failing key.

Generally the message of Cassandra is very verbose which is not uncommon in Java based applications that give a stack trace for debugging purposes.

Highlighting

Highlighting was done properly by both Elektra as well as Cassandra. Even though the log files of Cassandra do not have a highlighting integrated, the standard error message format which is common for Java applications makes it quite easy to apply highlighting. When we opened the log files via Visual Studio Code, the messages already got correctly highlighted which helped us to find the relevant information.

Consistent and Reduced Vocabulary

Table 4.15 shows the results of the vocabulary quality of Elektra vs. Cassandra.

Cassandra has consistent error messages but the majority of these messages use very specific vocabulary which comes from implementation details.

As an example take the following message from Cassandra which comes from setting the `thrift_prepared_statements_cache_size_mb` configuration setting to `true` (domain error) instead of a number:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

	Good	%	Bad	%
Elektra v0.9.0	121	89.63%	14	10.37%
Cassandra	27	16.98%	132	83.02%
Elektra v0.9.1	135	100.00%	0	0.00%

Table 4.15: Error message vocabulary of Elektra vs. Cassandra

```
ERROR [main] 2019-09-06 11:01:21,729 CassandraDaemon.java:749 -
    Exception encountered during startup: Invalid yaml: file:/
    home/wespe/.ccm/LyreBirdCluster/node1/conf/cassandra.yaml
Error: null; Can't construct a java object for tag:yaml.org
    ,2002:org.apache.cassandra.config.Config; exception=Cannot
    create property=thrift_prepared_statements_cache_size_mb for
    JavaBean=org.apache.cassandra.config.Config@64d2d351; For
    input string: "true"; in 'reader', line 1, column 1:
authenticator: AllowAllAuthenticator
^
... further stacktrace omitted
```

As an administrator or user without some advanced knowledge in Java the words `java object`, `JavaBean`, `org.apache.cassandra.config.Config@64d2d351`, `reader`, `AllowAllAuthenticator` etc. might be confusing. Cassandra though also has 27 good messages such as *ERROR [main] 2019-09-06 10:59:32,912 CassandraDaemon.java:749 - Insufficient permissions on directory /root.*

Elektra's bad vocabulary reactions come from the words `stat` and `normalize` which have been corrected in v0.9.1.

Personification and Apologizing

Cassandra did not personify any messages but could easily fix this issue for many error messages. Many error messages from Cassandra start with *Could not ...* or *Unable to ...*. Adding an `I` in front of the message (e.g., *I could not* would fix 1/3 of all messages).

The standardized error message format of Elektra personifies all messages as well as giving it an apologetic touch.

Internal Leakage

Table 4.16 shows the comparison of internal information leak of Cassandra vs Elektra.

One can see that this result for Cassandra is similar as with the vocabulary evaluation. The reason behind this is that many messages of Cassandra give a large stack trace which is obviously a violation of no internal leakages. These bad messages are primarily the incapability of the configuration parser to parse an incorrect type (e.g., `yalse` instead of `false`). The remaining custom error messages do not leak internals such as

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

	Good	%	Bad	%
Elektra	135	100.00%	0	0.00%
Cassandra	27	16.98%	132	83.02%

Table 4.16: Error message leakage of Elektra vs. Cassandra

```
ERROR [main] 2019-09-06 10:59:32,912 Directories.java:120 -
    Doesn't have execute permissions for /root directory
ERROR [main] 2019-09-06 10:59:32,912 CassandraDaemon.java:749 -
    Insufficient permissions on directory /root
```

Elektra did not leak internals of Cassandra at all due to the inaccessibility of internas.

4.4 Plugin Development

The specification language of Elektra already supports many validation features out-of-the-box such as regular expression checks, number ranges, types (String, Integer, float, etc.) or even checks for the existence of a certain directory, file. Some validation requirements though can be very specific and an additional extension of the specification language must be written. This section will investigate the effort needed to write such specification extensions.

We were able to track the time and effort needed for three additional extensions whereas two of them were written by ourselves.

The first extension we wrote addressed the need to check if a specific port is already occupied. If multiple applications run on a single system, it may happen that a desired port is already in use. This can easily happen if the same language framework such as Spring Boot is used because the default port might not have changed for one application. The writing of the extension took approximately 10 hours. The majority of the time can be allocated to discussions about the details such as metadata naming, should a retry also be included if a port is occupied, should a user or administrator be able to provide service port names too instead of numbers only, etc. [48]. Also some time was needed to get used to the ecosystem of Elektra, e.g., where and how to write tests, formatting and other build server requirements, correct places to alter or extend existing code, etc.

The second extension which we wrote validates permissions on directories or files. If an application for example wants to access certain files such as keystore files for SSL validation or data files on a network file system it might not have the appropriate permissions to do so. Also if the application has enough permissions to create and write log files to a directory could be an issue and already lead to latent errors in case of MySQL[33]. We needed around 27,5 hours to implement this extension. Compared to the first extension, the permission enhancement was considerably more complex to implement in the C language. There were many things to consider such as validation for a user

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

which does not perform the actual action (e.g., user `puppet` instead of actual user) or group permissions which required more tinkering in the code. The basic functionality including the discussions in GitHub took us 15 hours whereas requests to change behavior in the pull request took us 12,5 hours. Testing was also more difficult because temporary files had to be created on the build server as well as making the build stable even when run as root user on which permission checks make no sense.

The last extension was written by the Elektra developer Kodebach[23] who also tracked the effort and problems. The extension solves a specific problem for LCDproc which let users and administrators configure menu points on display devices. LCDproc gives the possibility to define menu points as configuration settings in such a way the they can reference submenues which themselves are configuration settings. However if a submenu references one of its parents it can run into an endless loop. The extension also checks if the parent menu references a correct submenu entry. This `reference` extension took Kodebach 24 hours whereas 4 hours were meetings for clarification of the extension scope and the remainder was implementation and further discussions on GitHub. Kodebach also noticed that the majority of the time needed is used for discussions.

4.5 Discussion

When writing the specification for both applications we clearly noticed how vital a good documentation is. We think though, that if a developer of an existing project with deeper knowledge into the application can be much faster when writing a specification. The most time consuming task then will most likely be the introduction training into Elektra. As stated earlier we recommend a text editor with multi-cursor and regular expression search as most of the time similar settings will have similar validation. We especially have seen this for Cassandra.

The most challenging task when writing a specification is the validation of configuration values which expect an arbitrary string (and no enumeration) we experienced. Either they are a path configuration setting which one needs to investigate the correct permissions needed for the application, a regular expression or a text which cannot be validated without a specific extension such as display text, username, password, class names, etc. Numbers can also be problematic when there is no information about their possible ranges or even worse in an unlimited case. In the case a number has no upper or lower bound, we had to use regular expressions because they do not come with a bound. Since C is a typed language we could not use any types such as integer or long for these cases since they are naturally bound by the language itself. A mismatch of types from the application and Elektra made a problematic case. A language such as Java does not support unsigned types but has configuration settings which expect positive integers. This was the case for Cassandra where the specification expects an unsigned long whereas java expects a positive long configuration value. If the user now would take such a high value that is between long and unsigned long, the specification will not catch the overflow error. Elektra as of now also would not suffice as standalone

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

validation for Cassandras configuration due to the lack of certain functionality. Cassandra is able to validate certain dependencies of configuration settings more easily like the need to enable certain configuration settings in order for other configuration settings to take effect (e.g., set `client_encryption_options.enabled` to `true` so that the changed `native_transport_port_ssl` configuration setting takes effect).

Using Elektra solely as configuration framework for Cassandra would be difficult due to default values which depend on the target systems hardware specification. Elektra does only support hardcoded default values as of now. Some requirements of Cassandra also would need highly specialized extensions such as a validation if a self written Java implementation fulfills the interface requirements. We think though that Elektra perfectly suits for LCDproc in terms of specification language features. Nearly every configuration setting could be associated with a validation. Elektra as well as LCDproc are both written in C which inherently makes it easier to validate configuration settings because there is no possibility for Java class implementation issues or type mismatches.

The ICS catch rate of Elektra vs. LCDproc surprised us because Elektra caught more than double (161) misconfigurations compared to LCDproc (74). The self written parser from LCDproc did not have many validations implemented. The library parser of Cassandra on the other hand caught a majority of ICS, even though error messages were particularly verbose. We also noticed that domain errors are comparably difficult to protect against because the user's view on the configuration setting is that far away from the actual meaning that an error message might not make sense anymore since it assumes that the user already has a similar understanding of the configuration setting. Another finding we got from our injections are the difficulties of configuration validation for structure errors. If users might copy paste sections of other configurations (e.g., a state-of-the-art configuration copied from a GitHub repository) and paste it under a wrong path in their applications configuration, they will not receive appropriate feedback. The application does not recognize these configuration settings and will take the working default values. Cassandra though was able to implement some validation and emitted messages such as

```
Please remove properties [commitlog_sync_period_in_ms,
   commitlog_sync, ____dirdata] from your cassandra.yaml
```

Cassandra though was not able to catch all of these misconfigurations. Path related errors were caught with one exception from both applications very well which Li et al.[8] said are difficult to diagnose. In our results the semantic errors have the highest failure rate.

Despite some varying success of the misconfiguration catch rate of Elektra, it clearly shines in regards of error message quality. With very few exceptions the message directly showed the ICS with an appropriate reason. When compared to Cassandra we noticed the benefit of a short and concise error message as Cassandra is overly verbose in many cases, probably attributed to how the Java ecosystem handles error messages. Elektra's error message system was carefully designed from the very start which was also suggested as a key aspect by Brown[1]. As a result all messages are consistent and easy to understand as

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

well as easy to change. This additional layer of feedback also has the benefit of prohibiting internal leakages (e.g., method names, class names, etc.) which are of no concern to the user. Also the use of highly specialized vocabulary is less likely to occur due to the extra layer not knowing about such specific details.

A combination of Elektra's validation and the applications complex configuration validation is a good approach to have the benefits of both worlds. One concern though which comes up are inconsistent error messages when mixing configuration validations. One could remedy this issue by writing an own validation extension for Elektra. If time is no hard constraint we would recommend this approach. Elektra also supports many language bindings which allows to write the extension in one's preferred language. This is especially favorable for complex validations which are related to programming language features such as interfaces, templates, etc. Writing an extension takes a fair amount of time when someone is not used to the concepts of Elektra. As a result it is also a decent solution to let Elektra catch the majority of the errors and let the application check for complex and specialized errors.

4.6 Implications - Improving Error Messages

Throughout our thesis we have seen many error messages and came across various problems. We also put a huge effort to harmonize our error message and code system of Elektra. There is though a great likelihood that any newly added message will again violate these standards. The only remedy as of now are good pull request reviews which again are prone to human errors. We want to present common pitfalls we have seen when correcting these messages and afterwards show potential remedies which we also have partly implemented ourselves.

4.6.1 Common Problems

1. Error messages are inconsistent

    a) Too advanced vocabulary

    Many error and warning messages included words which are too advanced for normal users. Examples are "stat", "normalize", "glob", uncommon acronyms, etc.

    b) Forgetting quotes around placeholders in strings

    In Elektra many error messages use variable arguments which are passed to a string formatter. It though is very confusing for a user if one of placeholders in the text contains spaces. As an example take the error message *Key {} with value {} is too long* where `{}` are placeholders. If now the user sets for example `hello_text="welcome.  Enjoy your day"` the resulting error message is not clear. As a result such placeholders should be surrounded by some symbol such as single quotes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

c) Sentence patterns are inconsistent

Many error messages violate the project's message design guideline such as sentences having to start with an upper case letter and should not contain a dot at the end for example. Exclamation marks are also non advisable since they are interpreted as shouting.

d) Leaking internals

We have seen error messages that leak internas such as method names and variable names in all of our investigated applications, including Elektra itself. An example is in LCDproc where the message *xosd: xosd_set_font() failed* clearly leaked an internal. Such internals though should be of no relevance for the user.

2. Forgetting crucial error information

   Crucial information is regarded as essential to the user for solving the underlying ICS. We have seen some generic error messages that are more confusing than helping such as *Could not rename file*. This message does neither state which file, is the file existing or does the user trying to rename just have insufficient rights, etc.

   Such information often lies in certain language specific containers such as `errno` in C or an error object in GoLang that can easily be forgotten to be added to the error message.

3. Forgetting the configuration key or configuration value information in error messages

   The by far most easy-to-forget information we have seen in error messages are the location of the ICS. This either means that configuration key or configuration value was missing.

4.6.2 Remedy

We want to present an approach where developers are forced to use a correct message when they have to use one.

"Architectural Design - Templates" The approach for dealing with such problems is a solid architectural design that uses templates. A template is predefined text with (optionally) placeholders in it and a different method signature to enforce optimal error messages. In Elektra there is already a central point of emitting error messages which can be enhanced to force developers into correct behavior. By using predefined templates for error messages we could see beneficial effects on the system.

For Elektra we have done this for out-of-memory error as an example [26]. Before we introduced such template there were a multitude of different non consistent wordings that are confusing for users. We changed the signature of the method which emits out-of-memory errors so that the text is predefined which is shown to the user. As a result we can easily fulfill all three common problems mentioned above.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

We also did this approach for semantic validation errors and found interesting side effects [27]. Especially for semantic validation errors we could see that problem 3 occurs very often. As a result we enforced the passing of configuration setting to the method so that at a central point we can guarantee that the configuration key and value is always present. The error message then takes the following form: *Key '{}' with value '{}' does not fulfill: {}* where `{}` are placeholders for the key, value and error reason. This approach had additional benefits:

- Good reevaluation of existing errors

  By the new signature we had to rework all existing error messages in that category and found that some errors were miscategorized just by seeing that no configuration setting could be given to the template. Some of these errors were categorized differently as a result. Some errors were also found to be better categorized into a new category which better suits its properties.

- Guiding developers to use the correct category

  Despite Elektra has a decent guideline on categorizing errors, a method signature is an additional compiler enforced guideline. If the developer may not find the appropriate arguments for the template he will most likely try to put an error into the wrong category.

Such templates can also deal with forgetting crucial error information which is available in error containers such as `errno` (problem 2). Elektra has its own category for resource errors such as missing files, insufficient permissions, non reachable hosts, etc. This category often uses library functions that set an `errno` which was forgotten by many developers. The template can easily set it without the need for the developer to provide it.

One disadvantage of this approach comes from duplicate information. If developers provide the same information in the open text area of the method signature (such as the error reason for the semantic validation template) the constructed message may be verbose and confusing. We think that the disadvantage of duplicate text does outweigh the costs of missing information.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 5

Conclusion

We have seen that a configuration specification for an application can have various success when it comes to catching misconfigurations. The more complex the application's configuration is, the less likely a validation can be applied from a general purpose configuration framework such as Elektra. Especially language features can be a problem such as inheritance in Java for Cassandra. For LCDproc, we have found that the majority of misconfigurations can be caught. For the database tool Cassandra, the specification could not catch as many errors due to configuration settings which are complex and not yet supported by Elektra. Examples are password configuration settings for key stores (JKS) or configuration settings which expect Java classes that implement certain interfaces.

The time needed to write specifications is coupled to the documentation provided for configuration settings as well as the possibilities for validation in the configuration language. For LCDproc with partly undocumented configuration settings but many possibilities for validation we needed about 25 hours. Cassandra on the other hand took us only about 8 hours due to good and detailed documentation and lack of validation possibilities.

Writing extensions for Elektra which solved some common mistakes in configurations took us between 10 - 27,5 hours. The large gap comes from the complexity of the extensions as well as getting used to the concepts and development of Elektra.

The way Elektra was designed around error messages resulted in far better messages for both applications. Elektra was able to correctly pinpoint the invalid configuration settings in the error message to 53% of LCDproc and 90% of Cassandra. LCDproc itself correctly pinpointed error messages to 24% whereas Cassandra achieved 17%. Especially compared to Cassandra, Elektra was able to give more succinct and encouraging error messages which frustrates user less likely. The additional layer of configuration validation also comes with advantages such as the avoidance of leaking internas of the application.

Since error messages are such a central aspect of good user experience and solution finding we identified common problems that occur when writing error messages. We conclude that text templates for error message can remedy these problems efficiently and also help developers to find the correct error category. We want to emphasize that a good architectural design around error message is essential for good system reactions and error messages as it enforces desirable properties from the very start.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Figures

2.1 Misconfiguration generation rules[8] 7
2.2 Misconfiguration diagnosis rate of different systems with three kinds of misconfiguration[8] 9

3.1 Top down view on how *Lyrebird* works with the application. 15
3.2 Top down view on how *Lyrebird* works with the Elektra. 17

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

2.1 Analyzed Systems[8] 8
2.2 System reactions results[8] 8

4.1 Server configuration setting type statistics of LCDproc v0.5.7 for 41 drivers 22
4.2 Configuration setting types statistics of Cassandra v3.11.4 23
4.3 LCDproc configuration injections per concrete error type 24
4.4 System reactions of Elektra vs. LCDproc. Elektra catches errors via a self written specification whereas the application catches errors via a configuration validation which is integrated into the application itself 25
4.5 Reactions of Elektra per Error Type 26
4.6 Reactions of LCDproc per Error Type 26
4.7 Error message quality of Elektra vs. LCDproc 29
4.8 Error message vocabulary of Elektra vs. LCDproc 30
4.9 Error message internal leakage of Elektra vs. LCDproc 31
4.10 Cassandra configuration injections per concrete error type 31
4.11 Reactions of Cassandra vs. LCDproc 33
4.12 Reactions of Cassandra per Error Type 33
4.13 Reactions of Elektra per Error Type 33
4.14 Error message quality of Elektra vs. Cassandra 34
4.15 Error message vocabulary of Elektra vs. Cassandra 36
4.16 Error message leakage of Elektra vs. Cassandra 37

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] P. J. Brown. „Error Messages: The Neglected Area of the Man/Machine Interface“. In: *Commun. ACM* 26.4 (Apr. 1983), pp. 246–249. ISSN: 0001-0782. DOI: 10.1145/2163.358083. URL: http://doi.acm.org/10.1145/2163.358083.

[3] T. Flowers, C. A. Carver, and J. Jackson. „Empowering students and building confidence in novice programmers through Gauntlet“. In: *34th Annual Frontiers in Education, 2004. FIE 2004.* Oct. 2004, T3H/10–T3H/13 Vol. 1. DOI: 10.1109/FIE.2004.1408551.

[4] Björn Hartmann et al. „What Would Other Programmers Do: Suggesting Solutions to Error Messages“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, pp. 1019–1028. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753478. URL: http://doi.acm.org/10.1145/1753326.1753478.

[5] Peng Huang et al. „ConfValley: A Systematic Configuration Validation Framework for Cloud Services“. In: *Proceedings of the Tenth European Conference on Computer Systems*. EuroSys '15. Bordeaux, France: ACM, 2015, 19:1–19:16. ISBN: 978-1-4503-3238-5. DOI: 10.1145/2741948.2741963. URL: http://doi.acm.org/10.1145/2741948.2741963.

[6] Lorenzo Keller, Prasang Upadhyaya, and George Candea. „ConfErr: A tool for assessing resilience to human configuration errors“. In: *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)* (2008), pp. 157–166.

[7] 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. Providence, Rhode Island, USA: ACM, 2011, pp. 109–116. ISBN: 978-1-4503-0829-8. DOI: 10.1145/2016911.2016934. URL: http://doi.acm.org/10.1145/2016911.2016934.

[8] S. Li et al. „ConfVD: System Reactions Analysis and Evaluation Through Misconfiguration Injection“. In: *IEEE Transactions on Reliability* 67.4 (Dec. 2018), pp. 1393–1405. ISSN: 0018-9529. DOI: 10.1109/TR.2018.2865962.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[9] X. Liao et al. „Do You Really Know How to Configure Your Software? Configuration Constraints in Source Code May Help“. In: *IEEE Transactions on Reliability* 67.3 (Sept. 2018), pp. 832–846. ISSN: 0018-9529. DOI: 10.1109/TR.2018.2834419.

[10] Guillaume Marceau, Kathi Fisler, and Shriram Krishnamurthi. „Mind Your Language: On Novices' Interactions with Error Messages“. In: *Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! 2011. Portland, Oregon, USA: ACM, 2011, pp. 3–18. ISBN: 978-1-4503-0941-7. DOI: 10.1145/2048237.2048241. URL: http://doi.acm.org/10.1145/2048237.2048241.

[34] S. Joon Park, Craig M. MacDonald, and Michael Khoo. „Do You Care if a Computer Says Sorry?: User Experience Design Through Affective Messages“. In: *Proceedings of the Designing Interactive Systems Conference*. DIS '12. Newcastle Upon Tyne, United Kingdom: ACM, 2012, pp. 731–740. ISBN: 978-1-4503-1210-3. DOI: 10.1145/2317956.2318067. URL: http://doi.acm.org/10.1145/2317956.2318067.

[35] Markus Raab. „Context-aware Configuration“. dissertation. Vienna: Faculty of Informatics, 2017. URL: https://publik.tuwien.ac.at/files/publik_265528.pdf.

[36] Markus Raab. „Improving system integration using a modular configuration specification language“. In: *MODULARITY*. 2016.

[38] Brian J. Reiser, John R. Anderson, and Robert G. Farrell. „Dynamic Student Modelling in an Intelligent Tutor for LISP Programming“. In: *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'85. Los Angeles, California: Morgan Kaufmann Publishers Inc., 1985, pp. 8–14. ISBN: 0-934613-02-8. URL: http://dl.acm.org/citation.cfm?id=1625135.1625137.

[39] T. Uchiumi, S. Kikuchi, and Y. Matsumoto. „Misconfiguration detection for cloud datacenters using decision tree analysis“. In: *2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. Sept. 2012, pp. 1–4. DOI: 10.1109/APNOMS.2012.6356072.

[40] T. Wang et al. „MisconfDoctor: Diagnosing Misconfiguration via Log-Based Configuration Testing“. In: *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. July 2018, pp. 1–12. DOI: 10.1109/QRS.2018.00014.

[42] John Wrenn and Shriram Krishnamurthi. „Error Messages Are Classifiers: A Process to Design and Evaluate Error Messages“. In: *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! 2017. Vancouver, BC, Canada: ACM, 2017, pp. 134–147. ISBN: 978-1-4503-5530-8. DOI: 10.1145/3133850.3133862. URL: http://doi.acm.org/10.1145/3133850.3133862.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[43] Tianyin Xu et al. „Early Detection of Configuration Errors to Reduce Failure Damage". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016, pp. 619–634. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/xu.

[44] Jiaqi Zhang et al. „EnCore: Exploiting System Environment and Correlation Information for Misconfiguration Detection". In: *SIGPLAN Not.* 49.4 (Feb. 2014), pp. 687–700. ISSN: 0362-1340. DOI: 10.1145/2644865.2541983. URL: http://doi.acm.org/10.1145/2644865.2541983.

[45] Sai Zhang and Michael D. Ernst. „Proactive Detection of Inadequate Diagnostic Messages for Software Configuration Errors". In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ISSTA 2015. Baltimore, MD, USA: ACM, 2015, pp. 12–23. ISBN: 978-1-4503-3620-8. DOI: 10.1145/2771783.2771817. URL: http://doi.acm.org/10.1145/2771783.2771817.

[46] Sai Zhang and Michael D. Ernst. „Which Configuration Option Should I Change?" In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: ACM, 2014, pp. 152–163. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568251. URL: http://doi.acm.org/10.1145/2568225.2568251.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Other references

[2] *diskd related 100% CPU after "squid -k rotate"*. https://bugs.squid-cache.org/show_bug.cgi?id=1703. Accessed: 2018-09-04.

[11] N/A. *'too many plugins' error*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2133 (visited on 10/13/2019).

[12] N/A. *boolean: accepted true/false values cannot be configured*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2466 (visited on 10/13/2019).

[13] N/A. *Cassandra*. July 2019. URL: https://github.com/apache/cassandra (visited on 06/15/2019).

[14] N/A. *ElektraInitiative*. July 2019. URL: https://www.libelektra.org/docgettingstarted/github-main-page (visited on 06/10/2019).

[15] N/A. *Error improvements*. Dec. 2019. URL: https://github.com/ElektraInitiative/libelektra/pull/2950 (visited on 12/29/2019).

[16] N/A. *error: Improved error message for range plugin*. Dec. 2019. URL: https://github.com/ElektraInitiative/libelektra/pull/3071 (visited on 12/29/2019).

[17] N/A. *Globbing in Metadata does not show all relevant metadata of subsetting until a value is set*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2448 (visited on 10/13/2019).

[18] N/A. *How to activate the boolean plugin in a specification file*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2451 (visited on 10/13/2019).

[19] N/A. *How to get the typechecker plugin into kdb*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2180 (visited on 10/13/2019).

[20] N/A. *How to set a negative value with kdb set*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2468 (visited on 10/13/2019).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[21] N/A. *ini plugin has problems with multiline metadata*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2467 (visited on 10/13/2019).

[22] N/A. *KConfig XT*. July 2019. URL: https://techbase.kde.org/Development/Tutorials/Using_KConfig_XT (visited on 07/17/2019).

[23] N/A. *Kodebach*. Oct. 2019. URL: https://github.com/kodebach (visited on 10/02/2019).

[24] N/A. *LCDproc*. July 2019. URL: http://lcdproc.omnipotent.net/ (visited on 06/10/2019).

[25] N/A. *network plugin does not allow 'localhost'*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2203 (visited on 10/13/2019).

[26] N/A. *Out of memory error template*. Nov. 2019. URL: https://github.com/ElektraInitiative/libelektra/pull/3194 (visited on 11/26/2019).

[27] N/A. *Out of memory error template*. Nov. 2019. URL: https://github.com/ElektraInitiative/libelektra/pull/3263 (visited on 11/26/2019).

[28] N/A. *Registry Value Types*. July 2019. URL: https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-value-types (visited on 07/16/2019).

[29] N/A. *The MySQL Test Framework*. Oct. 2018. URL: https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_MYSQL_TEST_RUN.html (visited on 06/07/2019).

[30] N/A. *type plugin causes failure for specifications*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2425 (visited on 10/13/2019).

[31] N/A. *Validation Regex Bug?* Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2881 (visited on 10/13/2019).

[32] N/A. *Yamlcpp treats integers as booleans*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/issues/2833 (visited on 10/13/2019).

[33] *No warn/error message if "log-error" is misconfigured (causing latent log loss)*. https://bugs.mysql.com/bug.php?id=74720. Accessed: 2018-09-04.

[37] Eric Raymond. *GPSD*. July 2019. URL: http://www.aosabook.org/en/gpsd.html (visited on 07/18/2019).

[41] Christopher Watson, Frederick W. B. Li, and Jamie L. Godwin. „BlueFix: Using Crowd-Sourced Feedback to Support Programming Students in Error Diagnosis and Repair". In: *Advances in Web-Based Learning - ICWL 2012*. Ed. by Elvira Popescu et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 228–239. ISBN: 978-3-642-33642-3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[47] Michael Zronek. *Lyrebird*. Oct. 2019. URL: https://github.com/ElektraInitiative/lyrebird (visited on 10/02/2019).

[48] Michael Zronek. *Plugin proposal discussion*. Oct. 2019. URL: https://github.com/ElektraInitiative/libelektra/pull/2169 (visited on 10/07/2019).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub