Elektra: universal framework to access configuration parameters

Markus Raab[1]

[1]TU Wien (TUW)

23 July 2016

Paper DOI: http://dx.doi.org/10.21105/joss.00044
Software Repository: https://github.com/ElektraInitiative/libelektra
Software Archive: http://dx.doi.org/10.5281/zenodo.200894

Summary

Today, most software invent their own configuration systems to dynamically load their configuration files at run-time. Up to now, it was difficult to integrate and specify the configuration across such applications. Elektra (Raab 2016a) provides a framework to bridge this gap (Raab 2010). In its essence, Elektra can be thought of a database that uses standard configuration files to persist data. Furthermore, a simple configuration specification language can be used to describe the data and its access to it.

Because of its generic nature, we cannot give an exhaustive list of what can be done with Elektra. The obvious cases, how developers and administrators should use Elektra to avoid configuration integration issues mentioned earlier, are described in Elektra's documentation. But it is ongoing research to find further use cases where these abstractions are useful. In this paper, we will give three concrete examples where Elektra has value to the research community.

Context Awareness

Currently, applications sometimes modify configuration values before using them. The reasons for such modifications can be called context, e.g., the number of CPUs, the current operating system, or the battery status [raab2016persistent]. The modifications within applications are problematic because it is not transparent for the user which configuration values the application actually will use.

We propose to move the logic that is responsible for determining configuration values into Elektra's specification language (Raab 2015b) (Raab 2016b) (Raab 2016c). This way, the user can query the up-to-date configuration values and get identical results to what the application will see. But even better, users can change the way context is taken into account easily.

Elektra allows us to intercept unmodified applications (by ''hijacking" calls to their configuration system) (Raab 2016c). For example, an application calls the C-function getenv() but actually retrieves a value from Elektra and not from an environment variable. This way we can make applications context aware that previously were not.

Validation

Developers often do not provide a way to validate configuration files (Raab 2015a). So administrators are forced to start applications to see if the configuration file is rejected.

We propose to move the validation from the applications to Elektra's specification language (Raab 2016b). Then every modification of the configuration files via Elektra gets automatically validated. This can be via an editor, a graphical user interface, or a web interface.

Furthermore, based on the specification language, we can generate valid and invalid configuration files. Such configuration files can be used to test the behavior of applications, e.g., injecting faulty configuration files to see if applications crash.

Code Generator

Applications today often have hand-written glue code to transform the strings received from configuration files to the variables used in the application. In Elektra a code generator (Raab 2015a) allows us to generate this code. Based on the configuration specification, Elektra provides methods with type-safe access to configuration values.

This simplifies writing new applications (Raab and Puntigam 2014) because we can also generate documentation and code to parse command-line options. But this technique can also be used to replace existing hand-written code.

References

Raab, Markus. 2010. "A Modular Approach to Configuration Storage." *Master's Thesis, Vienna University of Technology.*

———. 2015a. "Safe Management of Software Configuration." In *Proceedings of the Caise'2015 Doctoral Consortium*, 74–82. urn:nbn:de:0074-1415-4: http://ceur-ws.org/Vol-1415/. http://ceur-ws.org/Vol-1415/CAISE2015DC09.pdf.

———. 2015b. "Sharing Software Configuration via Specified Links and Transformation Rules." In *Technical Report from Kps 2015.* Vol. 18. Vienna University of Technology, Complang Group.

———. 2016a. "Elektra." http://www.libelektra.org.

———. 2016b. "Improving System Integration Using a Modular Configuration Specification Language." In *Companion Proceedings of the 15th International Conference on Modularity*, 152–57. MODULARITY Companion 2016. New York, NY, USA: ACM. doi:10.1145/2892664.2892691.

———. 2016c. "Unanticipated Context Awareness for Software Configuration Access Using the Getenv Api." In *Computer and Information Science*, edited by Roger Lee, 41–57. Cham: Springer International Publishing. doi:10.1007/978-3-319-40171-3_4.

Raab, Markus, and Franz Puntigam. 2014. "Program Execution Environments as Contextual Values." In *Proceedings of 6th International Workshop on Context-Oriented Programming*, 8:1–8:6. NY, USA: ACM. http://dx.doi.org/10.1145/2637066.2637074.