Functional Programming for Distributed Systems with XC (Artifact)

Giorgio Audrito
University of Turin, Italy

Roberto Casadei
University of Bologna, Cesena, Italy

Ferruccio Damiani
University of Turin, Italy

Guido Salvaneschi
Universität St. Gallen, Switzerland

Mirko Viroli
University of Bologna, Cesena, Italy

Abstract

In the paper "Functional programming for distributed systems with XC" we present XC, a programming language to develop the collective behaviour of homogeneous distributed systems while abstracting over concurrency, asynchronous execution, message loss, and device failures. The paper describes the design of XC, formalizes a core calculus for it, and shows that XC can effectively capture the logic of several distributed protocols and applications including gossiping, distributed summarization, information flows over self-healing communication structures, and self-organizing behaviours. Then, it discusses implementation, in a Scala and a C++ embedded domain-specific language (DSL), and provides evaluation through a case study in a smart city scenario, called SmartC. The reusable artifact described in this paper contains precisely those software projects: the Scala DSL, referred to as XC/Scala; the C++ DSL, referred to as XC/C++; and the SmartC implementation in both DSLs.

2012 ACM Subject Classification Software and its engineering → Distributed programming languages; Software and its engineering → Abstraction, modeling and modularity; Theory of computation → Functional constructs; Computing methodologies → Distributed programming languages

Keywords and phrases Distributed programming, Field Calculi, Scala DSL, C++ DSL

Digital Object Identifier 10.4230/DARTS.8.2.8

Funding This work was supported by the EU/MUR FSE REACT-EU PON R&I 2014-2022 (CCI2014IT16M2OP005), the Swiss National Science Foundation (SNSF, No. 200429), the Hessian LOEWE initiative emergenCITY, and the Ateneo/CSP "Bando ex post 2020".

Related Article Giorgio Audrito, Roberto Casadei, Ferruccio Damiani, Guido Salvaneschi, and Mirko Viroli, "Functional Programming for Distributed Systems with XC", in 36th European Conference on Object-Oriented Programming (ECOOP 2022), LIPIcs, Vol. 222, pp. 20:1–20:28, 2022.
https://doi.org/10.4230/LIPIcs.ECOOP.2022.20

Related Conference 36th European Conference on Object-Oriented Programming (ECOOP 2022), June 6–10, 2022, Berlin, Germany

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2022 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Scope

This artifact includes the XC implementations described in Section 5 of the companion paper [2], and the case study implementation described in Section 6 of the companion paper. Its purpose is to show that the domain-specific languages (DSLs) provide a working implementation of XC, to

Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

allow the reproduction of the case study, and to discuss the reusability of the developed tools and experimental setups for further research.

Specifically, the artifact supports the following claims of the paper:

C1) *RQ 1: the decentralised execution of the* XC *program on each device results in the desired collective behavior.* Evidence is provided by executing the simulations and plotting the results.

C2) *RQ 2: the overall behavior can be expressed by composing functions of collective behaviour that correctly combine thanks to alignment.* Evidence is provided by the means of C1) and inspecting the source code of the case study.

C3) The Scala DSL provides a working implementation of XC. Evidence is provided by the means of C1), C2), and by inspecting and executing XC/Scala programs.

C4) The C++ DSL provides a working implementation of XC. Evidence is provided by the means of C1), C2), and by inspecting and executing XC/C++ programs.

Reusability can be assessed by extending or reusing the XC/Scala and XC/C++ implementations, to develop new constructs, algorithms, and applications, as well as by extending or modifying the case study to experiment with new scenarios and algorithms.

2 Content

The artifact package consists of a compressed archive with the following:

- the source code and build infrastructure of XC/Scala, a DSL for XC embedded in the Scala programming language;
- the source code and build infrastructure of XC/C++, a DSL for XC embedded in the C++ programming language;
- the source code and build infrastructure for the SmartC case study, both in XC/Scala and XC/C++;
- the Markdown file of the artifact manual;
- a `LICENSE` file.

It also includes a ready-to-use virtual machine image (OVA file) with all the required dependencies.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact content is also available at the following repositories:

- XC/Scala [5]: `https://github.com/scafi/artifact-2021-ecoop-xc`
- XC/C++ [1]: `https://github.com/fcpp/fcpp`
- SmartC (Scala implementation) [4]: `https://github.com/scafi/artifact-2021-ecoop-smartc`

4 Tested platforms

The Scala-based artifacts leverage Gradle, which runs on all major operating systems and requires only a Java JDK version 8 or higher to execute. Tests have been carried out in Ubuntu 20.04 and Windows 10 on a Dell Xps with Intel Core i7-8550U CPU @ 1.80GHz × 8, 16Gb RAM, SSD disk. Notice that an Internet connection is needed to download artifact dependencies for compilation and execution of the case study. Regarding memory requirements, it is suggested to have at least 1GiB of free RAM.

The C++-based artifacts leverage CMake, which runs on all major operating systems and requires only a C++ compiler (GCC or CLANG) to execute. Tests have been carried out on a MacBook Pro with Intel Core i9 CPU @ 2.40 GHz × 8, 32Gb RAM, SSD disk. There are no connectivity requirements nor stringent memory requirements for the C++-based artifact.

The VirtualBox image has been tested with VirtualBox 6.0 and VMWare Workstation Player 16 under Ubuntu 20.04.3 LTS and Windows 10.

5 License

The artifact is available under the Apache 2.0 license.

6 MD5 sum of the artifact

a86d8900a526e46a96e36680e4e6a95b

7 Size of the artifact

5.5 GiB

A Related Work

The XC/Scala and XC/C++ implementations are based on the ScaFi language [6, 7] and FCPP language [1], respectively. The XC/Scala case study consists of a simulation built using the Alchemist meta-simulator [8] and its Alchemist/ScaFi integration [11]. The XC/C++ case study consists of a simulation built on the integrated FCPP simulator [3]. Another aggregate computing language which is not an internal but rather a standalone DSL is Protelis [9]. For a comprehensive coverage of related work the reader can refer to [10].

References

1 Giorgio Audrito. FCPP: an efficient and extensible field calculus framework. In *Proceedings of the 1st International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS*, pages 153–159. IEEE Computer Society, 2020. doi:10.1109/ACSOS49614.2020.00037.

2 Giorgio Audrito, Roberto Casadei, Ferruccio Damiani, Guido Salvaneschi, and Mirko Viroli. Functional programming for distributed systems with XC. In Karim Ali and Jan Vitek, editors, *36th European Conference on Object-Oriented Programming, ECOOP 2022, June 6-10, 2022, Berlin, Germany*, volume 222 of *LIPIcs*, pages 20:1–20:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. To appear. doi:10.4230/LIPIcs.ECOOP.2022.20.

3 Giorgio Audrito, Luigi Rapetta, and Gianluca Torta. Extensible 3d simulation of aggregated systems with fcpp. In *24th International Conference on Coordination Models and Languages, Proceedings*, Lecture Notes in Computer Science. Springer, 2022. To appear.

4 Roberto Casadei. scafi/artifact-2021-ecoop-smartc: v1.2, 2022. doi:10.5281/ZENODO.6538822.

5 Roberto Casadei. scafi/artifact-2021-ecoop-xc: v1.2, 2022. doi:10.5281/ZENODO.6538810.

6 Roberto Casadei, Mirko Viroli, Giorgio Audrito, and Ferruccio Damiani. FScaFi : A core calculus for collective adaptive systems programming. In *ISoLA (2)*, volume 12477 of *Lecture Notes in Computer Science*, pages 344–360. Springer, 2020. doi:10.1007/978-3-030-61470-6_21.

7 Roberto Casadei, Mirko Viroli, Giorgio Audrito, Danilo Pianini, and Ferruccio Damiani. Engineering collective intelligence at the edge with aggregate processes. *Eng. Appl. Artif. Intell.*, 97:104081, 2021. doi:10.1016/j.engappai.2020.104081.

8 Danilo Pianini, Sara Montagna, and Mirko Viroli. Chemical-oriented simulation of computational systems with ALCHEMIST. *J. Simulation*, 7(3):202–215, 2013. doi:10.1057/jos.2012.27.

9 Danilo Pianini, Mirko Viroli, and Jacob Beal. Protelis: practical aggregate programming. In Roger L. Wainwright, Juan Manuel Corchado, Alessio Bechini, and Jiman Hong, editors, *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pages 1846–1853. ACM, 2015. doi:10.1145/2695664.2695913.

10 Mirko Viroli, Jacob Beal, Ferruccio Damiani, Giorgio Audrito, Roberto Casadei, and Danilo Pianini. From distributed coordination to field calculus and

aggregate computing. *J. Log. Algebraic Methods Program.*, 109, 2019. doi:10.1016/j.jlamp.2019.100486.

11 Mirko Viroli, Roberto Casadei, and Danilo Pianini. Simulating large-scale aggregate mass with alchemist and scala. In *Federated Conference on Computer Science and Information Systems (FedCSIS)*, volume 8 of *Annals of Computer Science and Information Systems*, pages 1495–1504. IEEE, 2016. doi:10.15439/2016F407.