Designing Asynchronous Multiparty Protocols with Crash-Stop Failures (Artifact)

Adam D. Barwell
University of St. Andrews, UK
University of Oxford, UK

Ping Hou
University of Oxford, UK

Nobuko Yoshida
University of Oxford, UK

Fangyi Zhou
Imperial College London, UK
University of Oxford, UK

Abstract

We introduce Teatrino, a toolchain that supports handling multiparty protocols with crash-stop failures and crash-handling behaviours. Teatrino accompanies the novel MPST theory in the related article, and enables users to generate fault-tolerant protocol-conforming Scala code from Scribble protocols. Local types are projected from the global protocol, enabling *correctness-by-construction*, and are expressed directly as Scala types via the Effpi concurrency library. Teatrino extends both Scribble and Effpi with support for crash-stop behaviour. The generated Scala code is executable and can be further integrated with existing systems. The accompanying theory in the related article guarantees deadlock-freedom and liveness properties for failure handling protocols and their implementation. This artifact includes examples, extended from both session type and distributed systems literature, featured in the related article.

2012 ACM Subject Classification Software and its engineering → Source code generation; Software and its engineering → Concurrent programming languages; Theory of computation → Process calculi; Theory of computation → Distributed computing models

Keywords and phrases Session Types, Concurrency, Failure Handling, Code Generation, Scala

Digital Object Identifier 10.4230/DARTS.9.2.9

Funding Work supported by: EPSRC EP/T006544/2, EP/K011715/1, EP/K034413/1, EP/L00058X/1, EP/N027833/2, EP/N028201/1, EP/T014709/2, EP/V000462/1, EP/X015955/1, NCSS/EPSRC VeTSS, and Horizon EU TaRDIS 101093006.

Acknowledgements We thank the anonymous reviewers for their useful comments and suggestions. We thank Jia Qing Lim for his contribution to the Effpi extension. We thank Alceste Scalas for useful discussions and advice in the development of this paper and for his assistance with Effpi.

Related Article Adam D. Barwell, Ping Hou, Nobuko Yoshida, and Fangyi Zhou, "Designing Asynchronous Multiparty Protocols with Crash-Stop Failures", in 37th European Conference on Object-Oriented Programming (ECOOP 2023), LIPIcs, Vol. 263, pp. 1:1–1:30, 2023.
`https://doi.org/10.4230/LIPIcs.ECOOP.2023.1`

Related Conference 37th European Conference on Object-Oriented Programming (ECOOP 2023), July 17–21, 2023, Seattle, Washington, United States

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2023 Call for Artifacts and the ACM Artifact Review and Badging Policy.

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

1 Scope

The artifact presents Teatrino, a code generation toolchain supporting Multiparty Session Type (MPST) protocols with crash-stop failures and crash-handling behaviours. Teatrino is written in Haskell and implements both global and local types, including projection, from the related article.

Global types are derived from a subset of the Scribble syntax [2] accepted by νScr, extended to support our crash-handling model, which is consumed by Teatrino as input. Protocol-conforming Scala code is generated via projection to local types. Runtime types, as indicated in the related article, are not supported in Teatrino since these are not used when specifying protocols. Generated code uses an extended form of the Effpi concurrency library; although executable upon generation, the code can be extended and integrated with existing systems by the programmer.

The artifact contains protocol specifications for all examples presented in the related article. The artifact additionally includes dependencies and configuration files in order to facilitate the execution of generated code.

For more details, please consult Section 6 in the related article, Appendix G in the full version [1], and the `README` file in the artifact.

2 Content

The artifact is packaged as a Docker image, containing the source code of Teatrino, our tool, and our extended Effpi concurrency library. The artifact also includes the benchmarks used in the paper to evaluate our toolchain.

We enumerate the contents of the home user directory (`/home/mpst/`) below (* indicates an executable file):

- `Lib/Teatrino/`: contains the source code for our Teatrino tool. We use the Stack build system.
- `build.sbt`: is the Scala sbt build file used to compile and run the generated code.
- `effpi/`: contains the extended Effpi concurrency library. Note that references to authors and/or copyright holders are to *original* authors and/or copyright holders of the library.
- `examples/`: contains example protocols.
- `genAll.sh`*: generates code using Teatrino for all Scribble files in `effpi`.
- `project/`: configuration files used by `build.sbt`.
- `runScala.sh`*: script for running a single Scala file generated by Teatrino.

The home user directory may also contain the below subdirectories.

- `scala/`: default output directory for generated code, produced by Teatrino.
- `effpi_sandbox/`: used to run generated code, produced by `runScala.sh`.

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 is also available at: `https://zenodo.org/record/7974824`. The source files can be accessed at `https://github.com/adbarwell/ECOOP23-Artefact`.

4 Tested platforms

The artifact has been tested under Linux (Ubuntu 22.04.01) and macOS (Ventura 13.3.1, M2). In principle, it should be able to run under a correct installation of Docker.

5 License

The artifact is available under the MIT licence (`https://opensource.org/license/mit/`).

6 MD5 sum of the artifact

94cc09960ca3a9558cc30925291eca5d

7 Size of the artifact

1.3 GiB

A Additional Information

For additional information, readers are invited to consult the `README.md` file in the Docker image, which contains information on how to use the artifact. Alternatively, the `README` file is available online at `https://github.com/adbarwell/ECOOP23-Artefact/blob/master/README.md`.

B On Functionality and Reusability

The artifact is functional with respect to two aspects: *i)* Teatrino can be used to generate Scala code in negligible time to implement multiparty protocols, and *ii)* protocol specifications are included for all examples in the related article. Instructions for both code generation and running our benchmarks can be found in the aforementioned `README.md` file, and the example protocol specifications can be found in the `examples` directory. The raw results files in `Lib/Teatrino/benchmark` contain the data used to generate Fig. 13 in the related article. Whereas reported results were taken directly on the specified test machine, benchmarks taken within the Docker image may be subject to increased variance due to the small average generation times.

Teatrino consumes any valid Scribble protocol file and is therefore reusable. The accepted syntax of our Scribble variant is described in the `README.md` file, by which the user may write their own protocols. Custom protocols in `Lib/Teatrino/scribble/` can be benchmarked by adding the base filename (e.g. `a_PingPongAll`) of the file to the list declared by `allExamples` in `Lib/Teatrino/benchmark/Main.hs` on Lines 35–56. Generated code uses standard Scala, facilitating integration into novel or existing systems. We include our extended Effpi concurrency library and the `sbt` build and configuration files needed to run the generated code. Default values can be replaced with specific values to be sent between protocol participants and with the code necessary to select which branch to take when sending messages.

References

1 Adam D. Barwell, Ping Hou, Nobuko Yoshida, and Fangyi Zhou. Designing asynchronous multiparty protocols with crash-stop failures. *CoRR*, abs/2305.06238, 2023. `doi:10.48550/arXiv.2305.06238`.

2 Nobuko Yoshida, Raymond Hu, Rumyana Neykova, and Nicholas Ng. The scribble protocol language. In *8th International Symposium on Trustworthy Global Computing - Volume 8358*, TGC 2013, pages 22–41, Berlin, Heidelberg, 2014. Springer-Verlag. `doi:10.1007/978-3-319-05119-2_3`.