interface_	**Interacting Decentralized Transactional and Ledger Architecture for Mutual Credit**

WP4

Iterative Component and Field Testing

Deliverable D4.1

Report on Component Testing

Horizon 2020

Project funded by the European Commission

Information and Communication Technologies

FET OPEN Launchpad Project

Grant no. 754494

Contract Number: 754494

Project Acronym: INTERLACE

Deliverable No: D4.1

Due Date: 30/04/2018

Delivery Date: 22/01/2019

Author: Giuseppe Littera (SARDEX), Paolo Dini (UH), Eduard Hirsch (SUAS)

Partners contributed:

Made available to: Public

Versioning

Version	Date	Name, organization
1	16/11/2018	Paolo Dini (UH)
2	07/01/2019	Giuseppe Littera (SARDE), Paolo Dini (UH), Eduard Hirsch (SUAS)
3	22/01/2019	Paolo Dini (UH), Eduard Hirsch (SUAS)

Internal Reviewer: Maria Luisa Mulas (SARDEX), Egon Börger (UNI PASSAU)

Abstract

This deliverable is a stub. The project ran out of time and funds before we reached the final workpackage on testing. In this short report we only mention what the next steps, funded with own resources, are likely to be.

Table of Contents

Chapter 1

Synthesis

Giuseppe Littera, Eduard Hirsch and Paolo Dini

The component testing of the platform was performed only at an essential level, as reported in deliverable D3.2 [4], due to lack of time and funding. Although the CoreASIM model and the blockchain implementation both performed as required according to the basic tests, we did not have time or resources to develop a production-level framework to test the components of the system formally and in an automated way against the requirements.

Over the next year we will investigate how Cucumber,[1] together with other testing methods, can be utilised to achieve a test plan which can be executed transparently and repeatedly. Cucumber is a tool which supports Behaviour-Driven Development (BDD [5]), which can be seen as an advancement relative to plain Test-Driven Development (TDD [1]) that adds "Deliberate Discovery" to the process. Deliberate Discovery can be summarized by the aim to "deliberately seek and discover what development teams are ignorant about before implementations starts". This goal/intention is very similar to the AS(I)M approach taken by INTERLACE, in other words the aim to get the implementation details right and clear for everyone.

In spite of some similarities, AS(I)Ms and Cucumber achieve this goal in rather different ways. Whereas AS(I)Ms rely on a very rigorous mathematical approach, Cucumber uses a natural language approach called Gherkin[2] connected with actual test implementations (in various languages[3]). The aim of the work will be to combine such methods with the AS(I)M approach, thereby giving the strict mathematical definitions a connection to broadly accessible and understandable terms which can easily be verified by both (non-)technical readers as well as by automated tests. This more immediate development of component testing will be performed with own funds as a collaborative effort between SARDEX, SUAS, UH, and the open source community at large. We expect the component testing to be completed sometime in 2019. All updates will be shared on GitHub[4] and/or on the project's website.[5]

An interesting possible future PhD-level research project could be to investigate the feasibility of developing an ASIM-to-Gherkin compiler in order to run acceptance tests with Cucumber directly from the ASIM model implementation. If feasible, it could ultimately be integrated in the CoreASIM framework. The motivation is that although ASIMs are mathematically rigorous they are completely flexible in how the universe of terms of a given model is defined. In fact, the standard ASM practice is to use natural language-like expressions to model any system [3, 2]. Since also Gherkin is flexible in its use of natural language expressions, it should be in principle possible to tailor ASIMs to the Gherkin/Cucumber test logic and natural language properties. A compiler that maps the specification logic to Gherkin would then achieve the best of both worlds: both mathematical rigour and human readibility at the specification, modelling, and testing phases.

[1] https://docs.cucumber.io/
[2] https://docs.cucumber.io/gherkin/
[3] Java Example: https://docs.cucumber.io/guides/10-minute-tutorial/#see-scenario-reported-as-undefined
[4] https://github.com/InterlaceProject/InterlaceBlockchain
[5] https://www.interlaceproject.eu/

References

1. Kent Beck. *Test-driven development: by example.* Addison-Wesley Professional, 2003.
2. E Börger and A Raschke. *Modeling Companion for Software Practitioners.* New York: Springer-Verlag, 2018.
3. E Börger and R Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis.* New York: Springer-Verlag, 2003.
4. E Hirsch, T Heistracher, P Dini, E Börger, L Carboni, M L Mulas, and G Littera. *D3.2: Final Demonstrator Implementation.* INTERLACE Deliverable, European Commission, 2018. URL: https://www.interlaceproject.eu/.
5. Matt Wynne, Aslak Hellesoy, and Steve Tooke. *The cucumber book: behaviour-driven development for testers and developers.* Pragmatic Bookshelf, 2017.