aleksander.es

Unit testing DBus services

Unit testing DBus services

aleksander.es

- Unit tests
- Library unit tests
- DBus client library * unit tests

(*) and service

Unit tests: introduction

aleksander.es

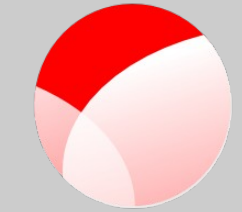
- Testing individual units of source code
  - E.g. single library/program functionalities.
- *White box* testing of the library/program
  - But each functionality treated as a *black box*.
- Main benefits:
  - Find problems early
  - Facilitates changes
  - Simplifies integration
  - Forces a good design

Unit tests: limitations

aleksander.es

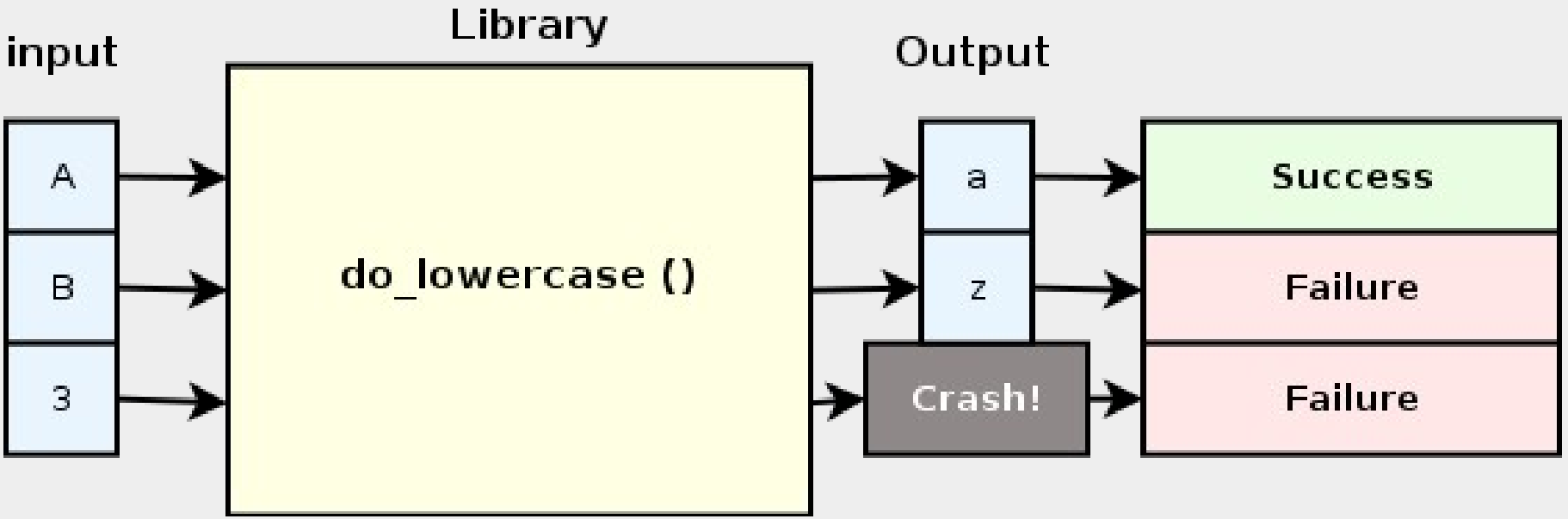
- Must be deterministic
- Must be isolated
- Hard to make good real-life situation tests
- Hard to make tests with external dependecies

Library unit tests

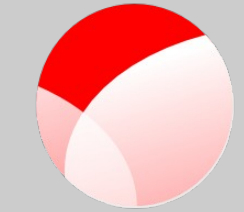


aleksander.es

Unit tests without context

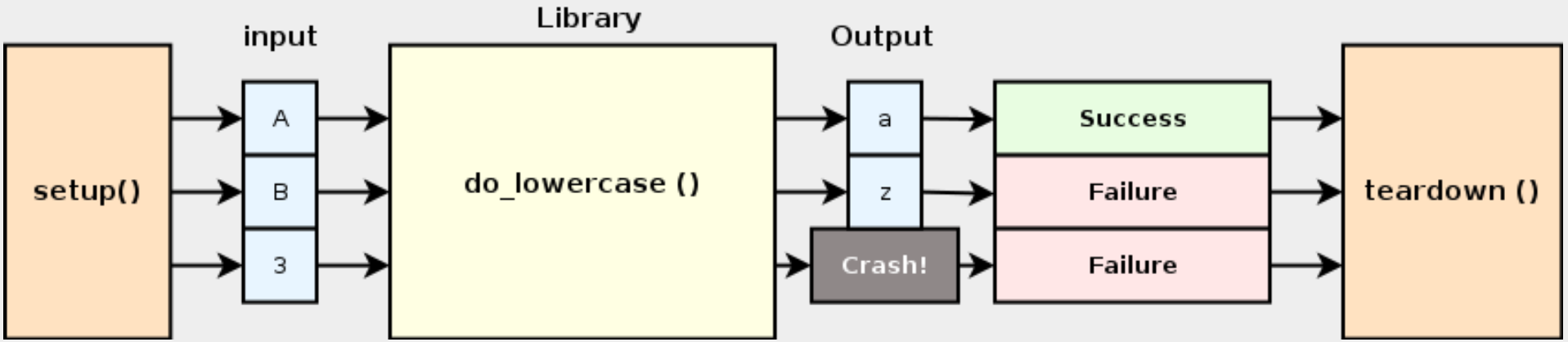


Library unit tests

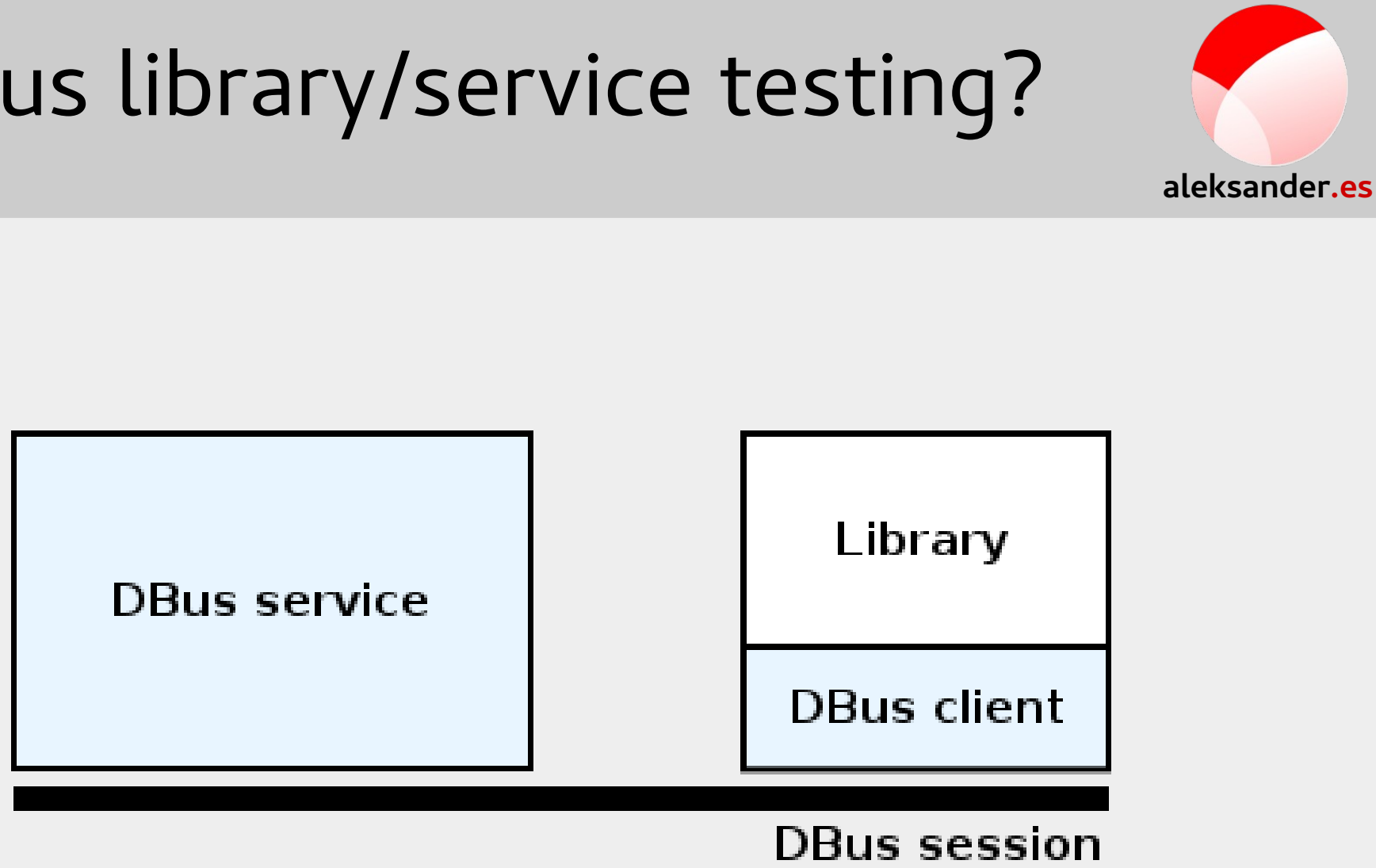


aleksander.es

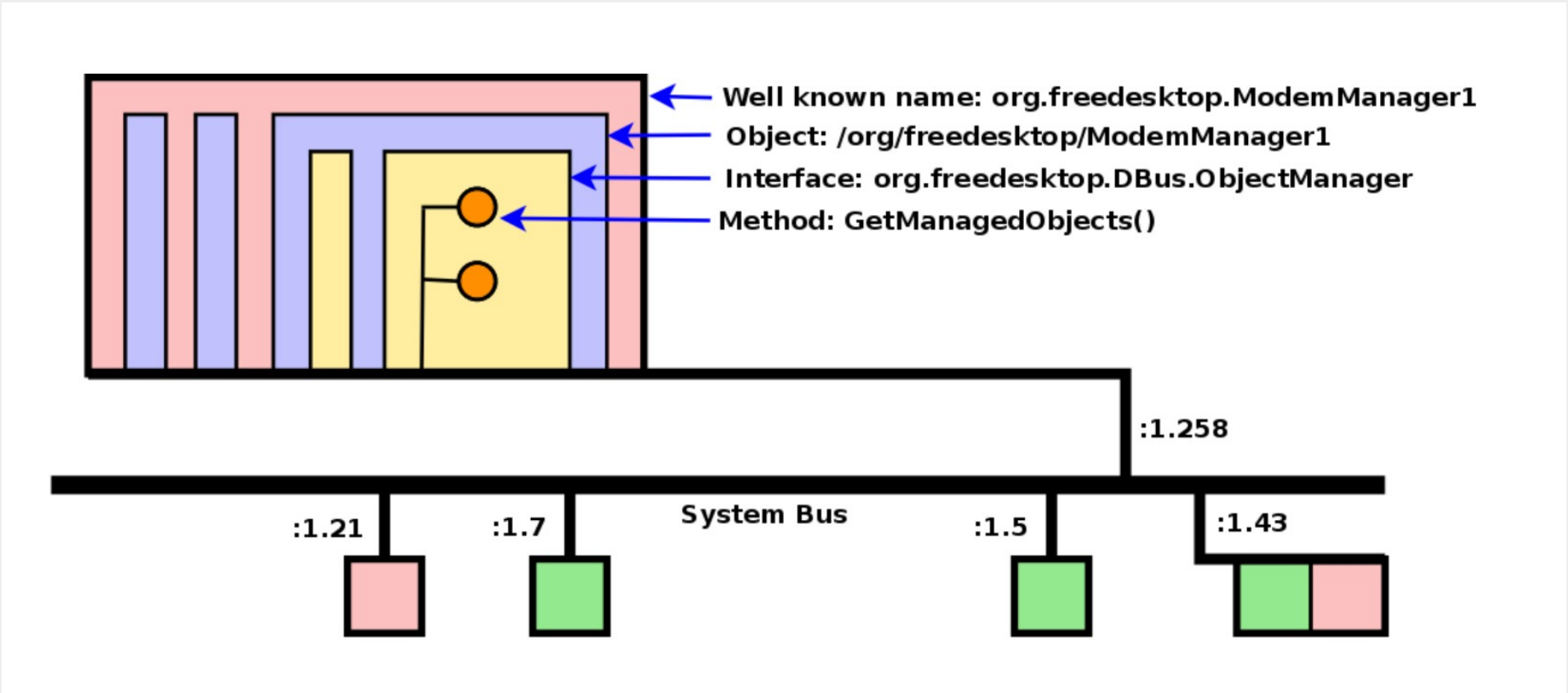
Unit tests with a common context: setup & teardown



DBus library/service testing?



DBus services vs clients

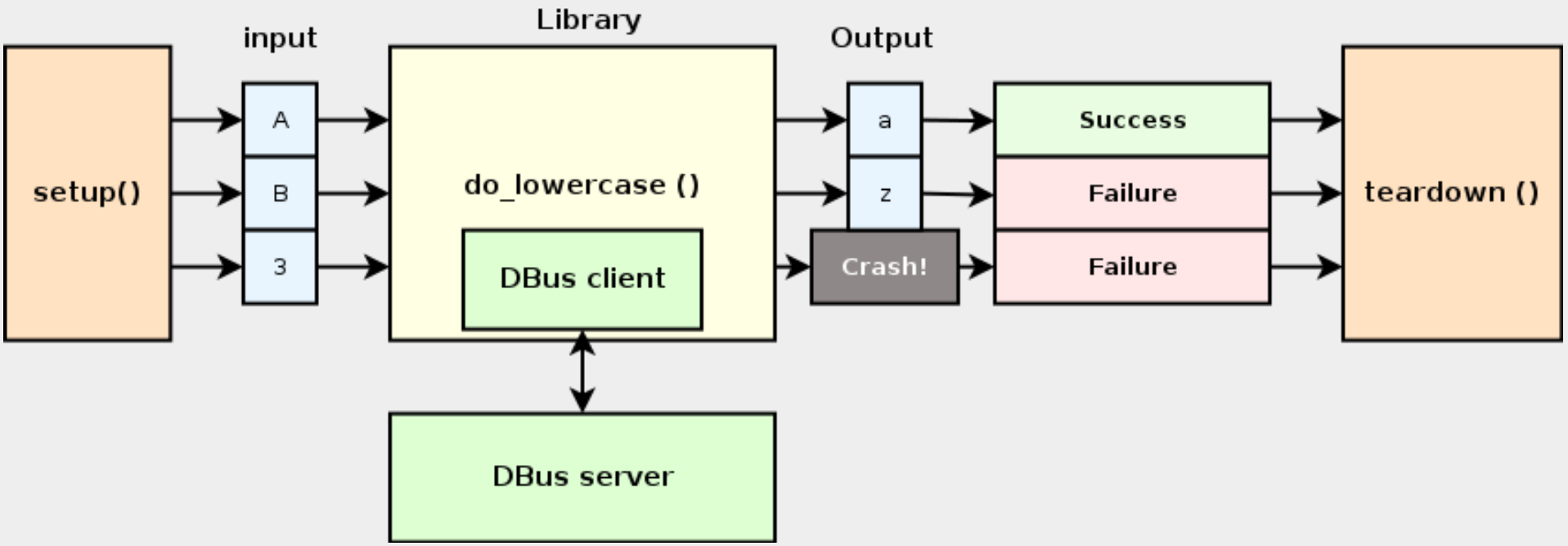


GTestDBus

aleksander.es

- Private DBus session running in the context of the unit test., since GLib/GIO 2.34.
- On test setup:
  - **g_test_dbus_up()**
- On test teardown:
  - **g_test_dbus_down()**
- Start the service:
  - **org.freedesktop.DBus.Peer.Ping()**

DBus based unit tests

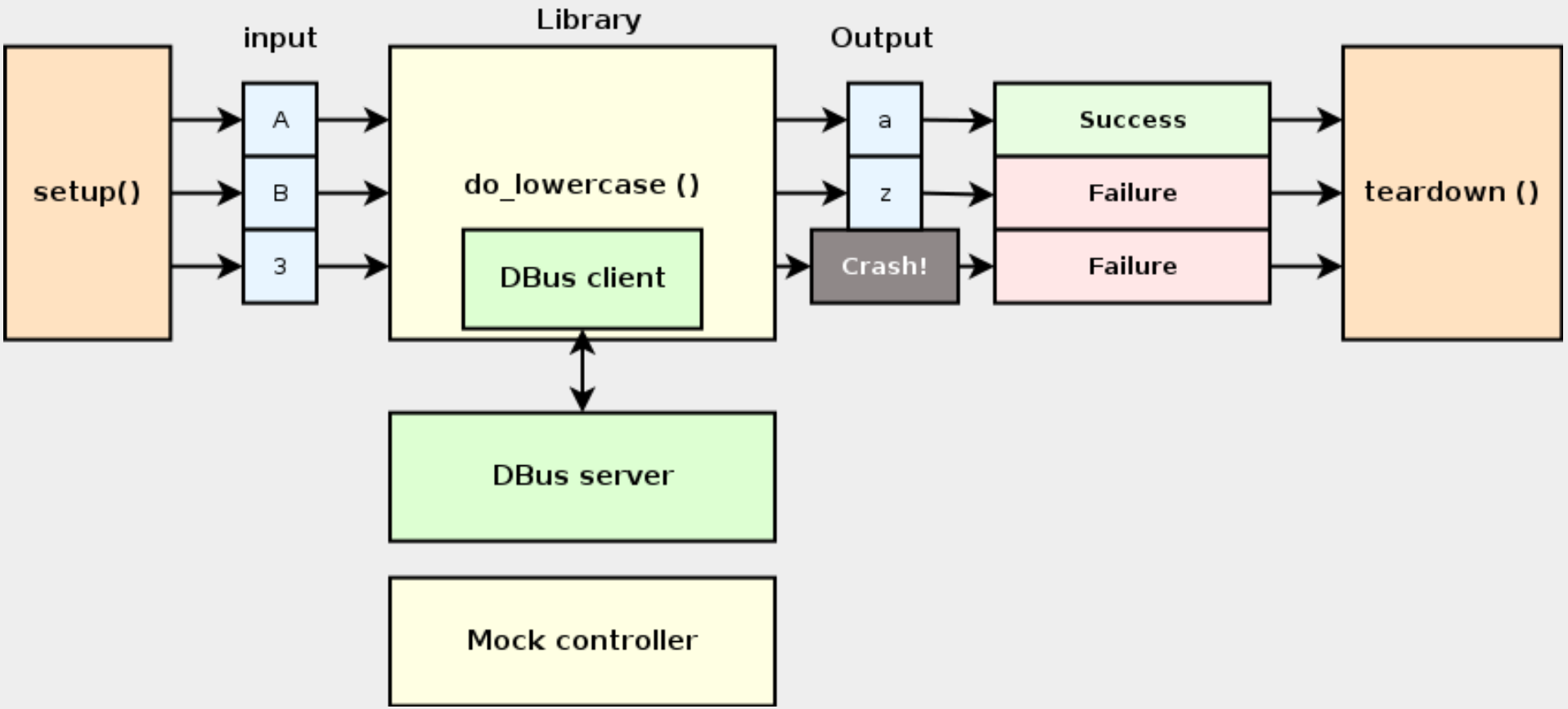


But...

aleksander.es

- The DBus service itself has functionality that may need to be simulated for the purposes of the test!
  - E.g. the behaviour of the ModemManager daemon depends on having actual modems available.

Mocking service logic



https://sigquit.wordpress.com/2014/02/14/simulating-2g3g4g-modem-behavior-in-modemmanager-tests

Thanks!

aleksander.es

+Aleksander Morgado

Freelance GNU/Linux developer

aleksander@aleksander.es

@aleksander0m

http://aleksander.es