Model-based Testing

experiences from practice

Marten Sijtema

Enschede, The Netherlands

UNIVERSITY OF TWENTE.

Part 1

Case-study

Part II

Model

Part III

SUT, JTorX, Adapter

Part IV

Recipe & Conclusions

Part 1

Case-study

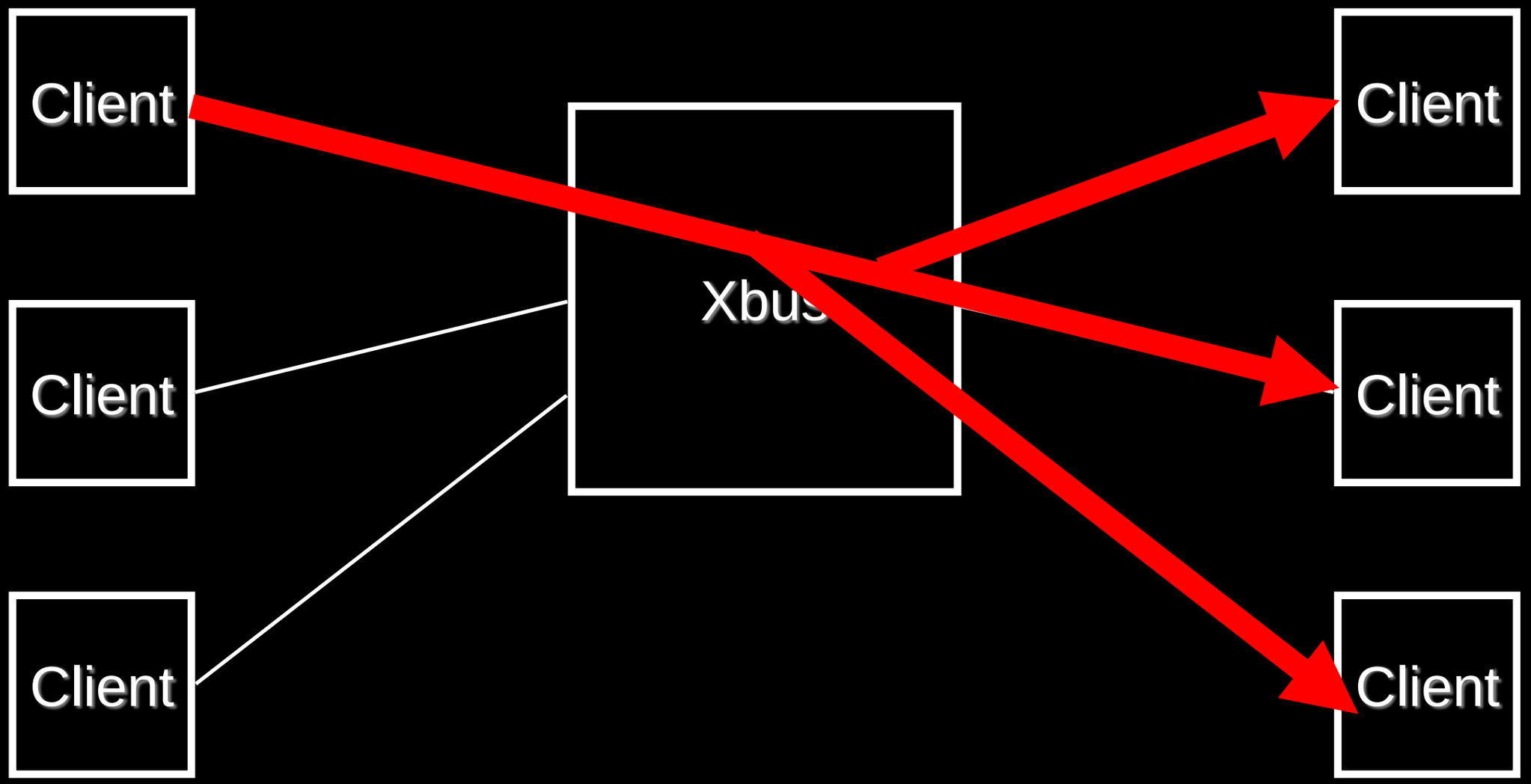
case study

at **Neopost**

(Austin, Texas, US)

UNIVERSITY OF TWENTE.

XBus



Possible clients

software

G2 | Navigator | Fusion

hardware

XBus Protocol

defined by user

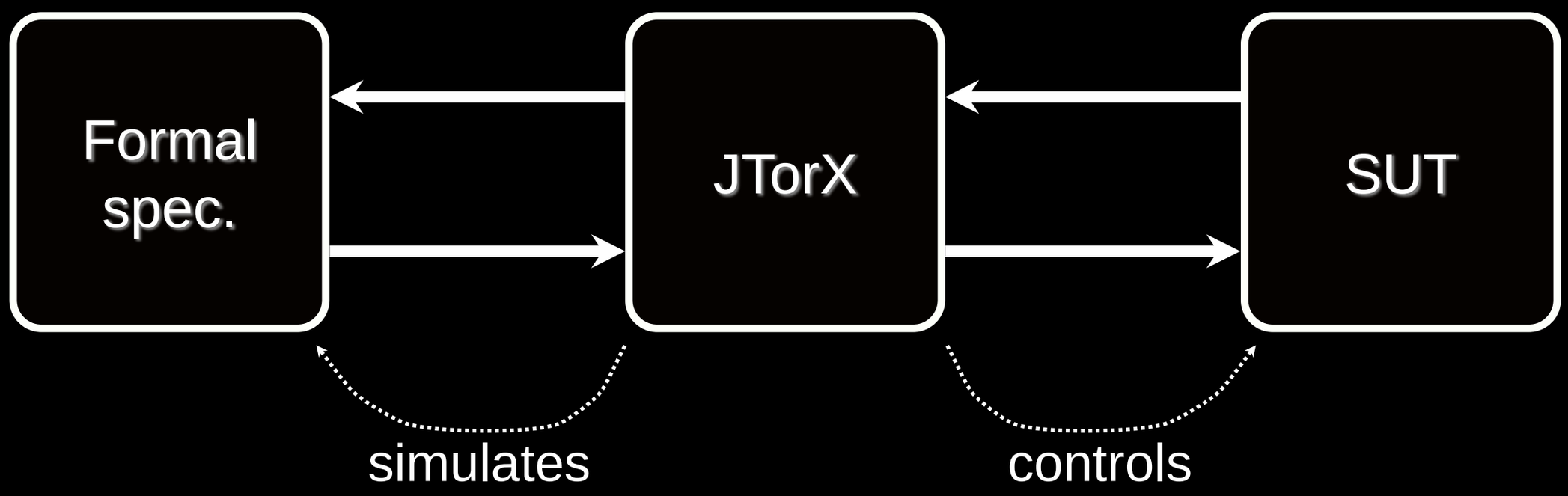
Application messages

Protocol messages

service advertisement,
service subscription,
service inquiry,
etc.

test XBus

model-based testing



Part II

Model

Written in **mCRL2**

Formal spec.

Describes **behaviour** of XBus

= XBus Protocol

= XBus messages + handling

Requirements

for model-based testing

- SUT input-enabled
- SUT logs output

I/O in our case

<Message />

<Message />

<Message />

our model

main_loop =

...

Connect +

ConnectAuthenticate +

Subscribe +

Advertise +

...

our model

main_loop =

...

Connect . **ConnectAcknowledge** +

ConnectAuthenticate +

Subscribe +

Advertise +

...

our model

main_loop =

...

Connect . ConnectAcknowledge **(id)** +
ConnectAuthenticate **(id)** +
Subscribe **(id, service)** +
Advertise (...) +
...

our model

```
main_loop =
            ...
            Connect . ConnectAcknowledge (id) +
            ConnectAuthenticate (id) +
            Subscribe (id, service) +
            Advertise (...) +
            ...
clients_array = struct Client (...)
...
```

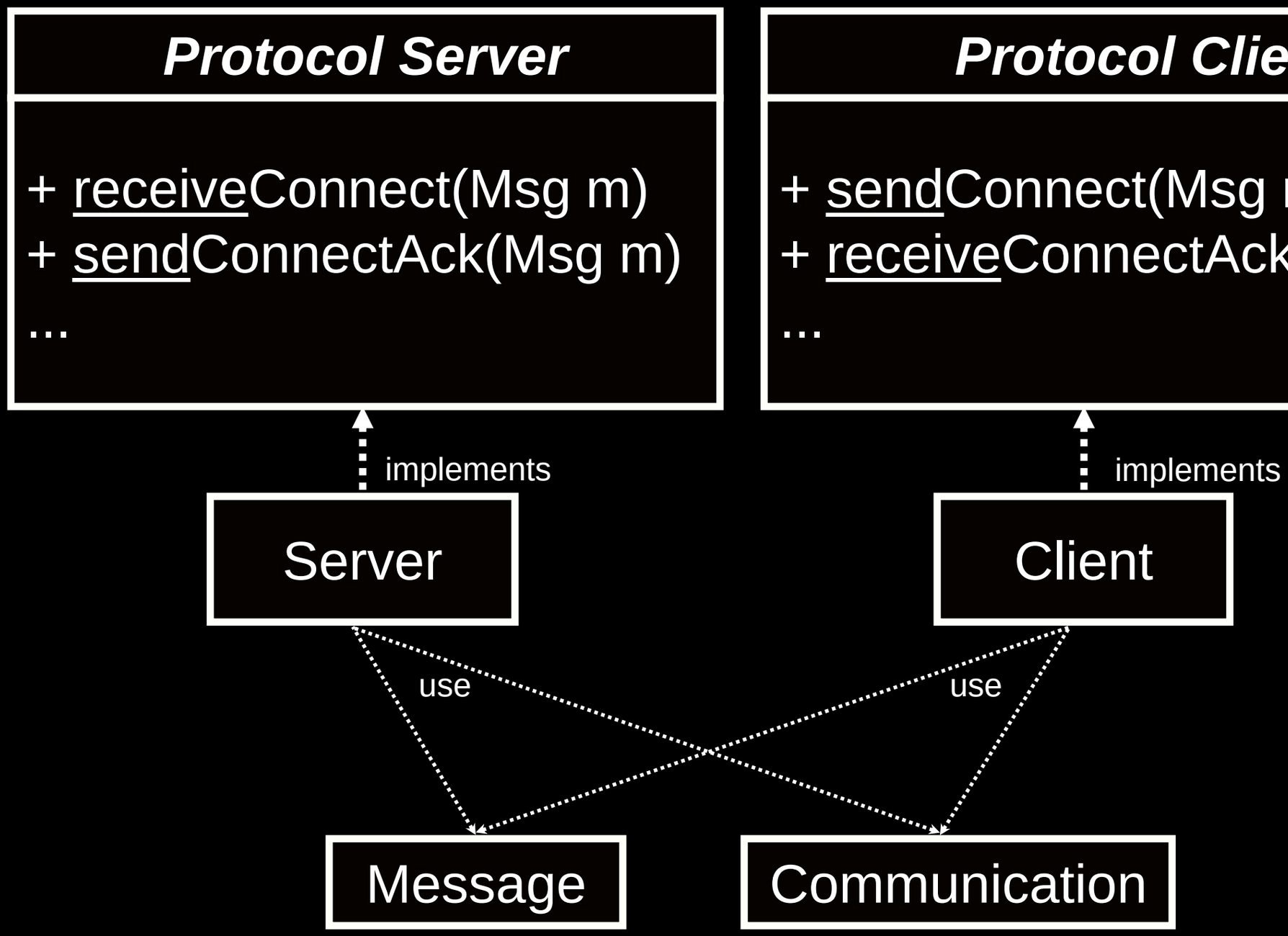
180 lines

UNIVERSITY OF TWENTE.

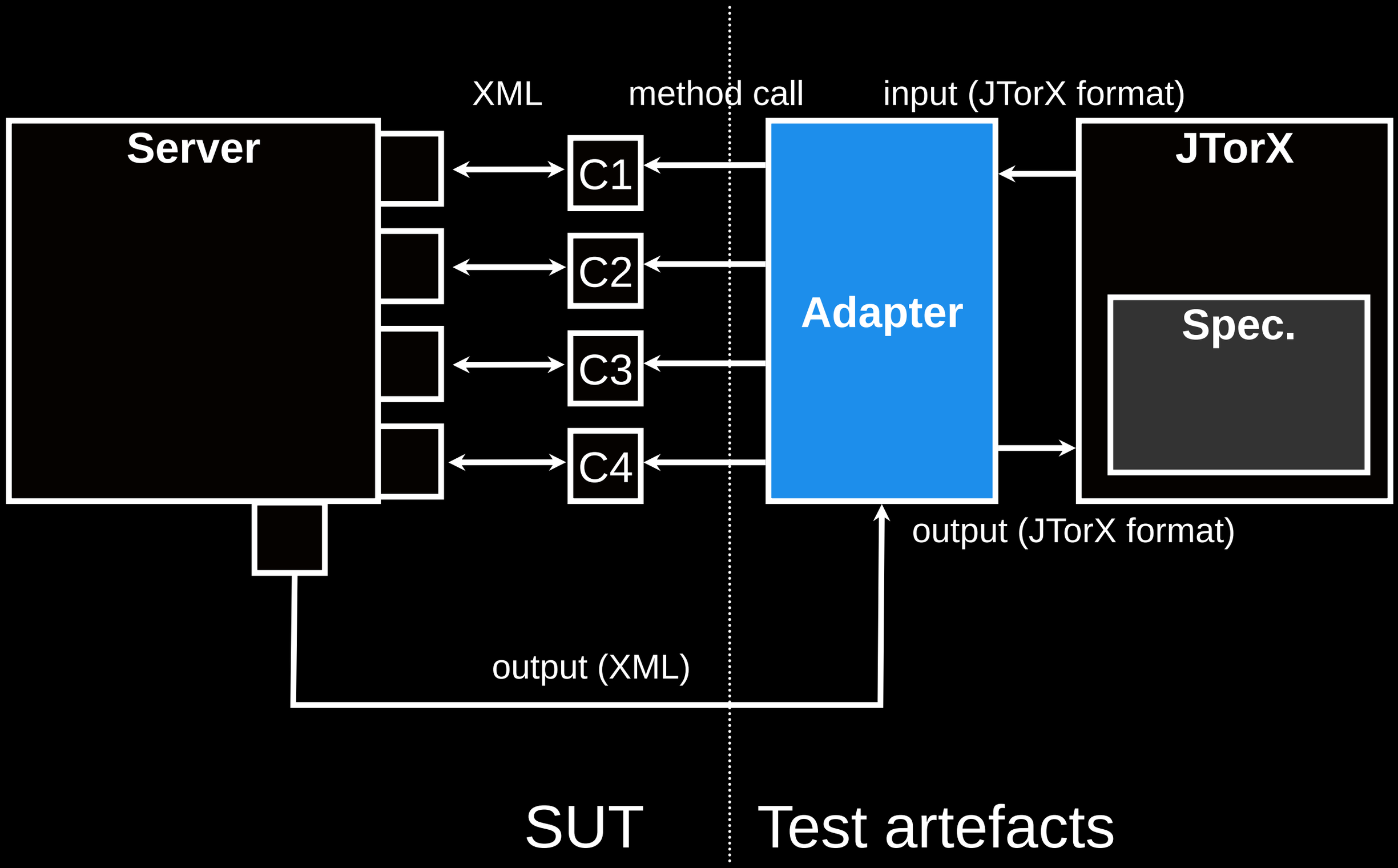
Part III

SUT, JTorX, Adapter

architecture



test architecture



receiving output

sendSubscribe(‘MyService’)

Client 2

Adapter

Subscribe!2!MyService

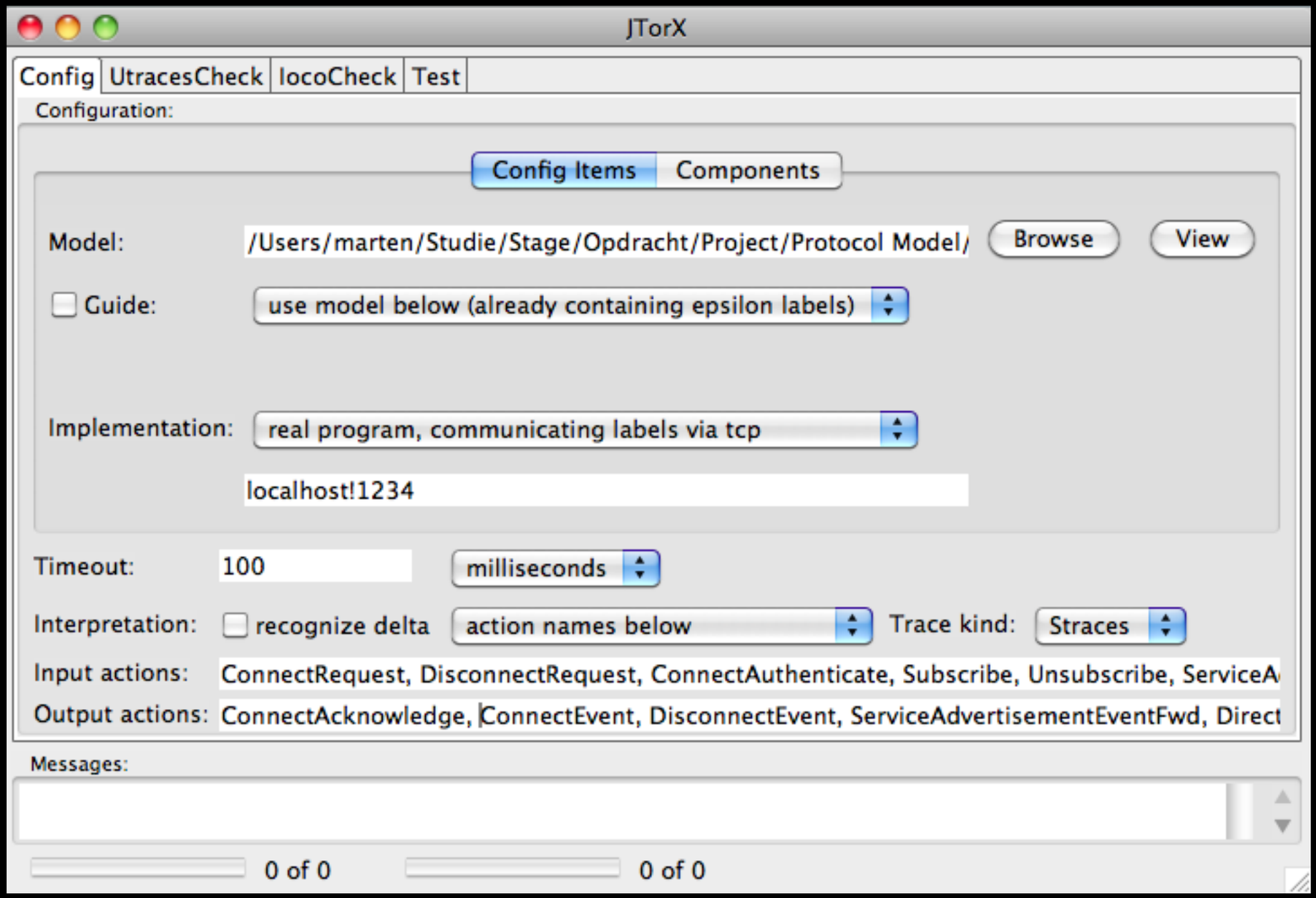
JTorX

ConnectAck!2

Server

<Msg type=”ConnectAck” sender_id=”2”>
...
</Msg>

run it!


JTorX
Config
UtracesCheck
IocoCheck
Test
Configuration:
Config Items
Components
Model:
/Users/marten/Studie/Stage/Opdracht/Project/Protocol Model/
Browse
View
Guide:
use model below (already containing epsilon labels)
Implementation:
real program, communicating labels via tcp
localhost!1234
Timeout:
100
milliseconds
Interpretation:
recognize delta
action names below
Trace kind:
Straces
Input actions:
ConnectRequest, DisconnectRequest, ConnectAuthenticate, Subscribe, Unsubscribe, ServiceA
Output actions:
ConnectAcknowledge, ConnectEvent, DisconnectEvent, ServiceAdvertisementEventFwd, Direct
Messages:
0 of 0
0 of 0

Part IV

Recipe & Conclusions

bugs found

- subtle ones
- hard to find without MBT

recipe

- define inputs and outputs
- make **SUT** input-enabled
- log output
- same rationale in **model** and **SUT**
- prepare architecture!

---

simple adapter

recipe@thema

- model security/transaction logic
- what is input?
- what is output?
- how to give input to SUT?
- how to log output from SUT?
- write adapter

conclusion

17%

thank you