github.com/cape-town-testing


JUMO'S TESTING STACK

PRINCIPLES AND PATTERNS

OPTIMISE FOR FAST FEEDBACK WHERE IT NEEDS TO BE FAST

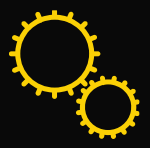
ASSERT THINGS THAT MATTER

BUILD FOR TESTABILITY

HUMANS NEED TO EXPLORE

VISIBILITY

NO WALL, NO THROWING THINGS OVER



GOOD UNIT TESTS

CUSTOMER CENTRIC SCENARIO TESTS RUN AS PART OF THE BUILD (BROADSTACK)

TOOLS TO ENABLE HUMAN INTERACTION & VISIBILITY OF APPLICATION

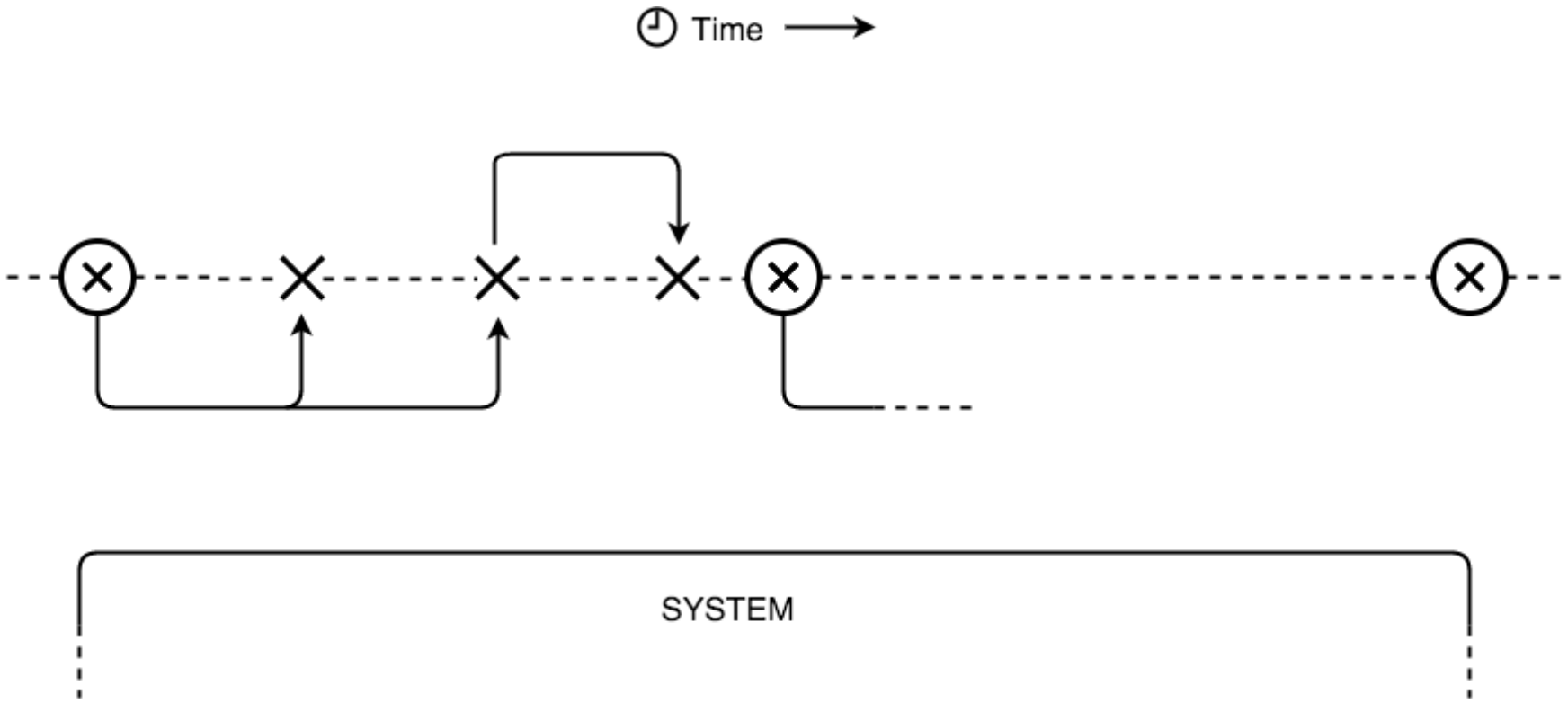
TOOLS TO REDUCE DEPENDENCY ON ENGINEERS FOR COMPLEX SCENARIOS

F O R T U N E
C O O K I E

"FORTUNE COOKIE"

SCENARIO DRIVEN LOAN-LIFECYCLE TESTING

- Provides a **(timeline)** testing **framework**.
- Setup a **scenario surrounding a customer** through lifetime events and **iterate through each one**.
- An event represents **a point in time that we will navigate** to and make **assertions** on.
- Operates at a **high level**.
- Instead of simply **testing multiple components** through integration tests, this **tests using behavioural patterns**.
- **Runs everytime we do a deploy / build of our system.**



Enqueued event (Notification, repayment etc)

Initialisation event (Cycle change)

Product code: _____					
Status: disbursed		Penalty fee: ____			
No. instalments: _		Instalment period: __		Loan term: 90	
Principal amount: ______		Total service fee: _____		Instalment amount: _____	

_DAY	_BALANCE	_ARREARS	_CD	_DESCRIPTION	_METADATA
0	______	_______	___	Task trigger	service_fee => _____, penalty_fee => _

- 1b -

_DAY	_BALANCE	_ARREARS	_CD	_DESCRIPTION	_METADATA
5	______	_______	___	Loan cycle change	
5	______	_______	___	Task trigger	service_fee => _, penalty_fee => _
5	______	_______	___	Repayment scheduler	
14	______	_______	___	One third instalment reminder	status => sent
15	_____	_______	___	Manual repayment	success, amount => _____
24	_____	_______	___	Two thirds instalment reminder	status => not sent
34	_____	_______	___	Imminent instalment reminder	status => not sent

- 2a -

_DAY	_BALANCE	_ARREARS	_CD	_DESCRIPTION	_METADATA
35	_____	_______	___	Loan cycle change	
35	_____	_______	___	Task trigger	service_fee => _____, penalty_fee => _
35	_____	_______	___	Repayment scheduler	
35	_____	_______	___	Scheduled repayment	failed, not permitted
35	_____	_______	___	Scheduled repayment	failed, not permitted
35	_____	_______	___	Scheduled repayment	failed, not permitted
35	_____	_______	___	Scheduled repayment	failed, not permitted
36	_____	_______	___	Penalty fee warning	status => not sent

- 2b -

_DAY	_BALANCE	_ARREARS	_CD	_DESCRIPTION	_METADATA
37	_____	_______	___	Loan cycle change	
37	_____	_______	___	Task trigger	service_fee => _, penalty_fee => _
37	_____	_______	___	Repayment scheduler	
46	_____	_______	___	One third instalment reminder	status => sent

BUGS UNCOVERED

- **Implementation error on scheduled job**
  - No unit / integration test coverage.
  - Alluded to by unexpected behaviour on our debug log.
  - Found: invalid method usage.
- **Notification (SMS) handler errors**
  - Multiple branching code paths. Complex test cases.
  - Each customer scenario tests multiple notification paths.
  - Found: incorrect notification timing and missing variables.

FURTHER USAGE

- **Debug log** used as a **teaching tool.**
  - Actual values and relative timing of events.
  - High level view.
- Provides insight into product behavior off the back of new changes.
  - Prototyping of new product constructs.

T I M E T R A V E L

TIME TRAVEL

THE PROBLEM

- Requirement for exploratory tests, particularly when testing or interacting with multiple UIs.
- Fixtures or mock data often make assumptions about how operations mutate and store data.
- As a financial services platform, we have many time dependent tasks, such as scheduled payments, interest and fee calculations.

TIME TRAVEL

THE SOLUTION

- We created a tool that gave us the ability to modify the current time in our system, allowing us to perform operations as though we were on a specific date or time.
  - Freeze time in an environment (across multiple processes)
  - Fast forward events and batch jobs
- This allowed us to simulate real customer / system interactions, using actual interfaces like USSD screens or back office system screens.
- This helped us identify consistency issues ahead of time, before rolling out to production.

TIME TRAVEL

BENEFITS AND TAKEAWAYS

- Early identification of consistency issues across UIs.
- This led to more rapid enhancements to our APIs and integration points.
- We could perform / test complete customer journeys using the actual system, ie. without relying simulations or assumptions about the environment or data.
- Testing and UAT could be extended beyond the engineering team, to non-technical stakeholders.
- Encouraged us to write code that is more testable from a time travel point of view.

T H A N K S

github.com/cape-town-testing

Cape Town Testing

Q3

17 August at **DVT**

Q4: 19th October

github.com/cape-town-testing