MORSE AND STATE-MACHINE-BASED TESTING FOR THE HUMANOID ROBOT HEAD FLOBI

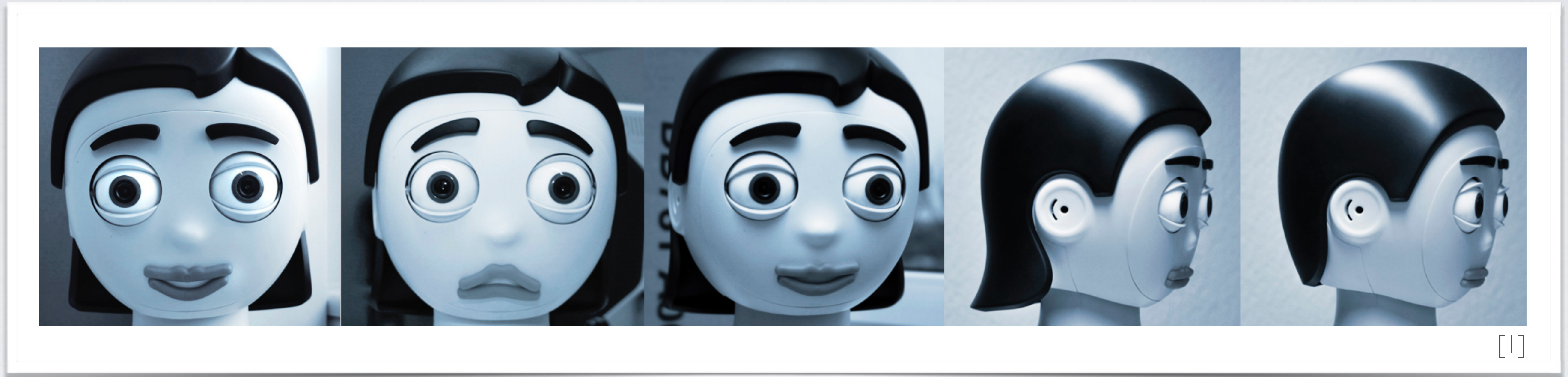
International Workshop on **MORSE** and **HRI** 2014

Florian Lier | CITEC, Bielefeld University | flier@cit-ec.uni-bielefeld.de | 07.03.2014

FLOBI

https://www.youtube.com/watch?v=dG7kNhxrOG8

FLOBI FACTS


[1]

Designed as a comic-like human face to avoid uncanny effects

Hair, eyebrows, lips and frontal face parts are exchangeable

Features stereo vision, stereo audio and a gyroscope

Innovative magnetic actuation mechanism

Custom design motor control boards

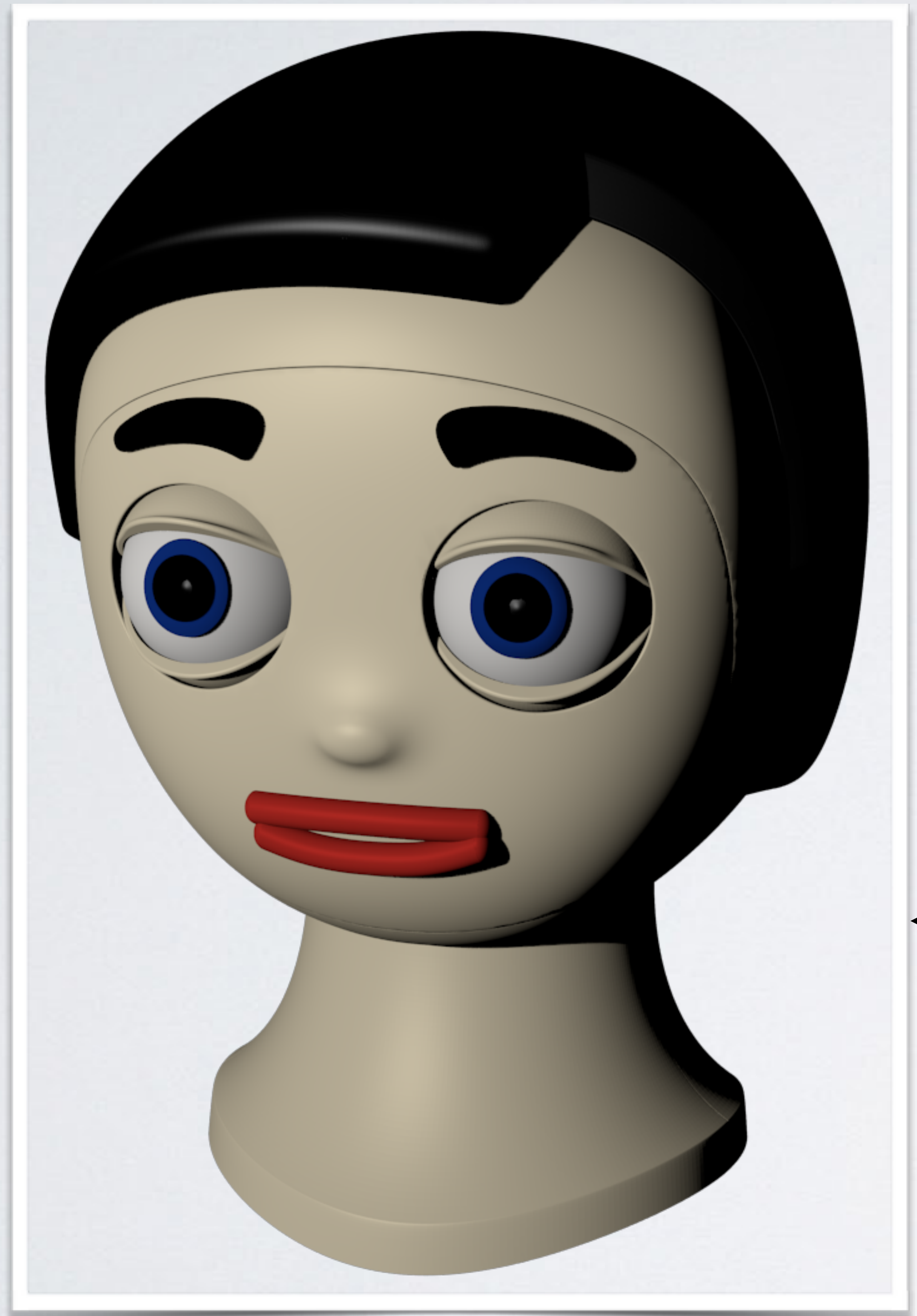
Eye saccades reach ~400 degree/s

Overall18 DoF

More about Flobi: http://pub.uni-bielefeld.de/publication?ftext=flobi

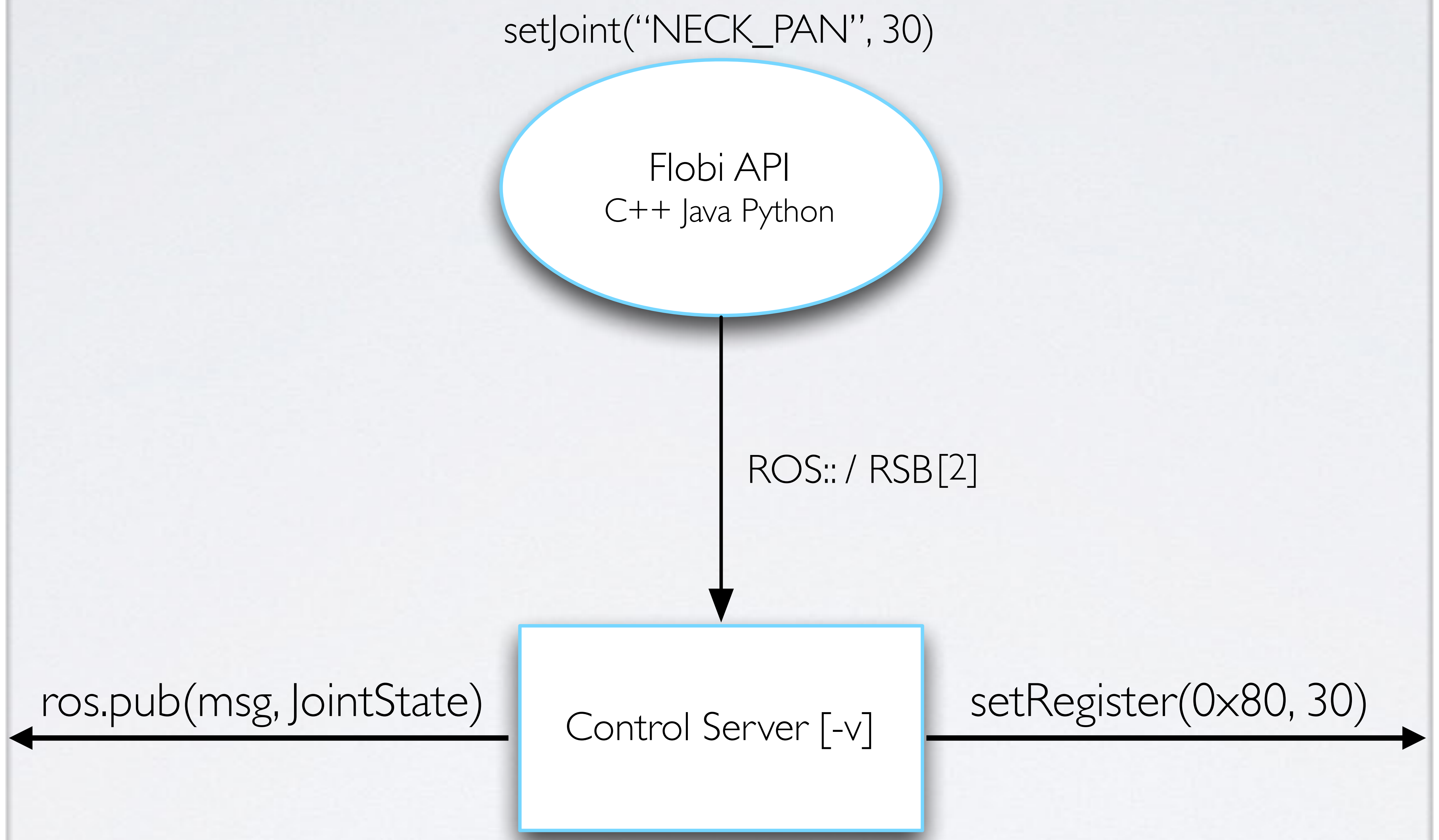
Rapid prototyping & testing required a simulator

FLOBI SOFTWARE STACK



Virtual Robot

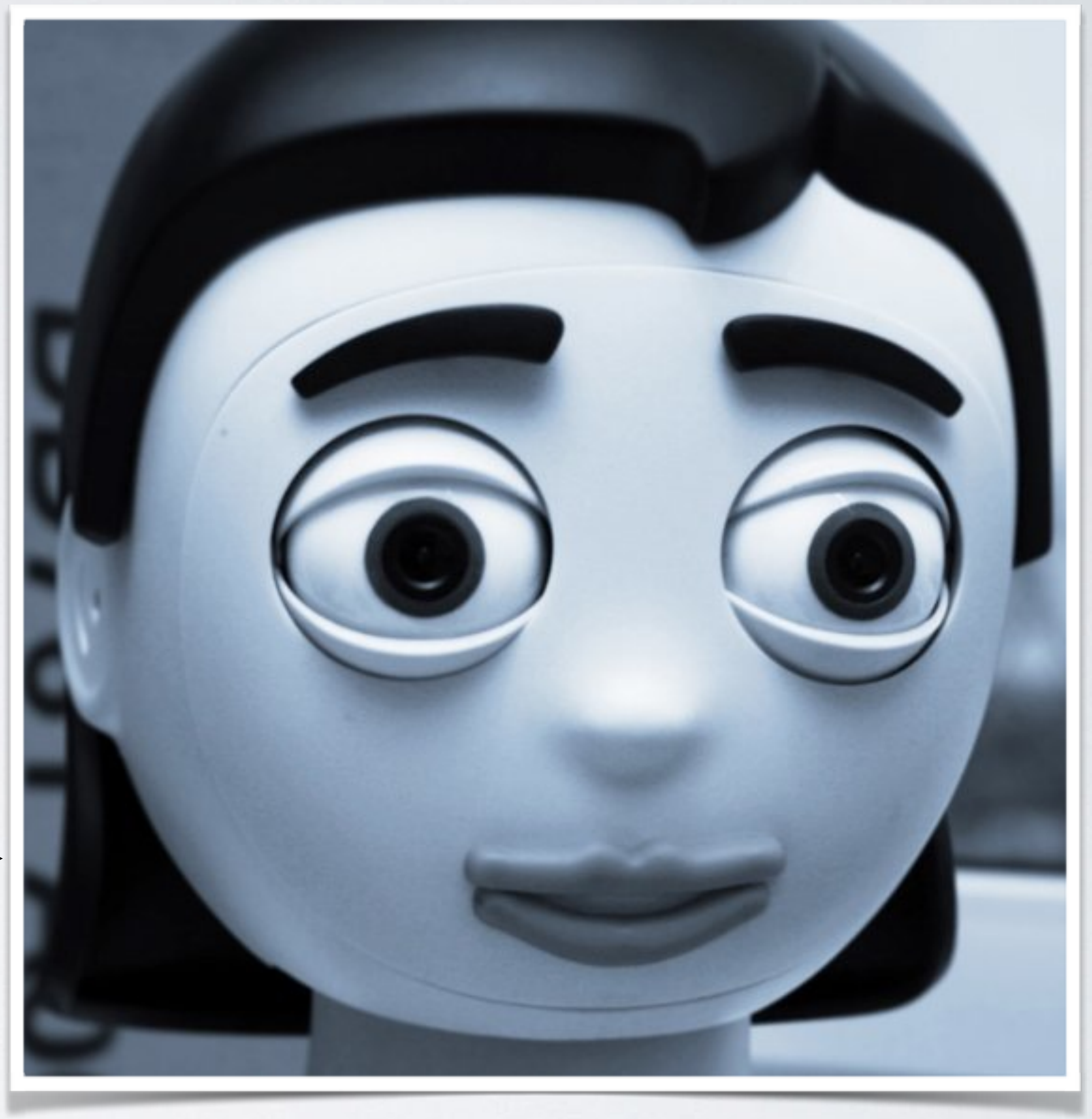
MORSE [5] (of course)



running in virtual or hardware mode

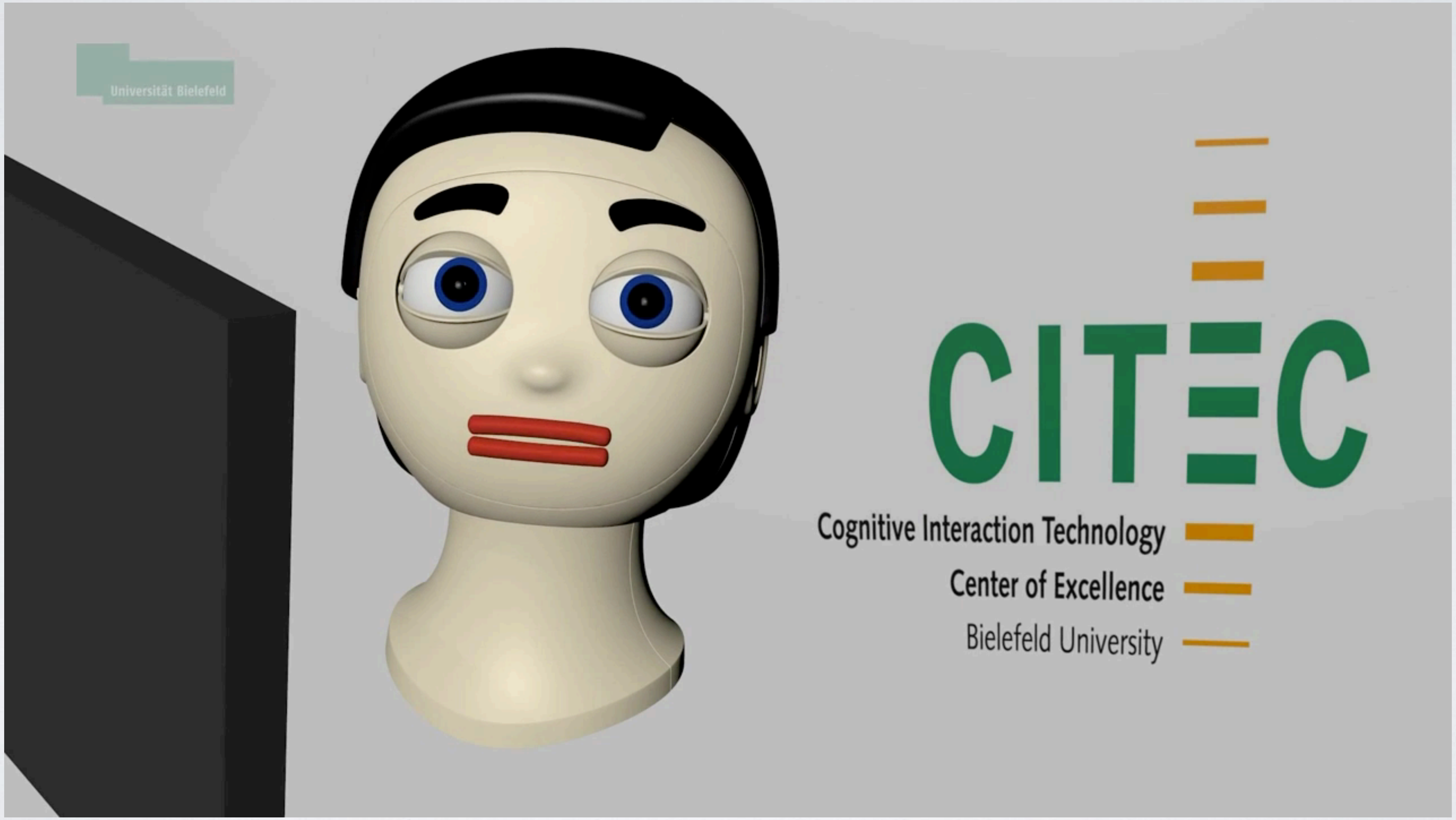
virtualizes controller boards in "-v mode"

simulates motor characteristics (ramps, vel, etc.)



Physical Robot

FLOBI SOFTWARE STACK



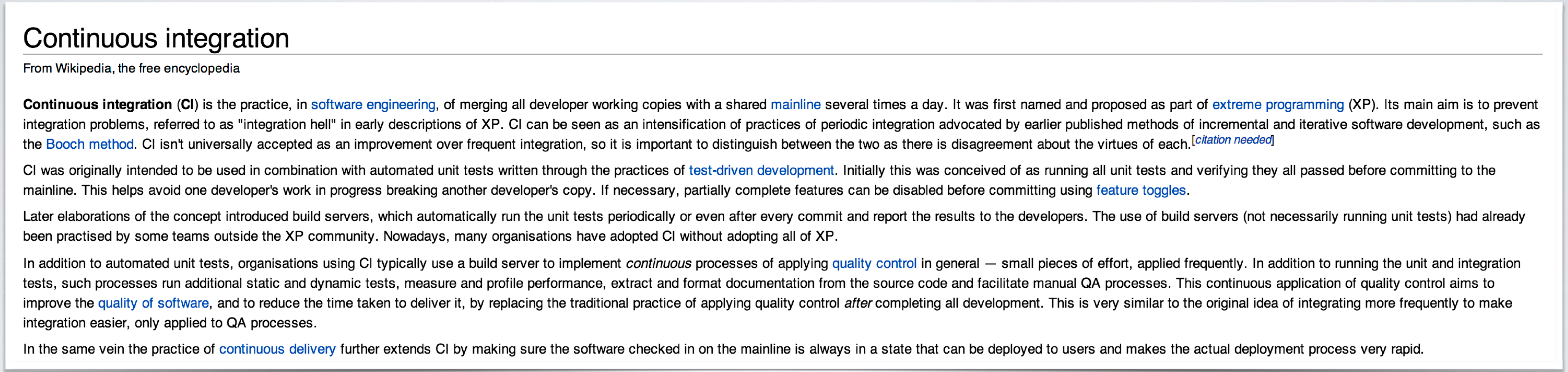
https://www.youtube.com/watch?v=PBs0c2LzMVM

But…

… how precise is our simulator?

/off: who doesn’t know CI?

CI PRIMER



Continuous integration

From Wikipedia, the free encyclopedia

Continuous integration (**CI**) is the practice, in software engineering, of merging all developer working copies with a shared mainline several times a day. It was first named and proposed as part of extreme programming (XP). Its main aim is to prevent integration problems, referred to as "integration hell" in early descriptions of XP. CI can be seen as an intensification of practices of periodic integration advocated by earlier published methods of incremental and iterative software development, such as the Booch method. CI isn't universally accepted as an improvement over frequent integration, so it is important to distinguish between the two as there is disagreement about the virtues of each.[*citation needed*]

CI was originally intended to be used in combination with automated unit tests written through the practices of test-driven development. Initially this was conceived of as running all unit tests and verifying they all passed before committing to the mainline. This helps avoid one developer's work in progress breaking another developer's copy. If necessary, partially complete features can be disabled before committing using feature toggles.

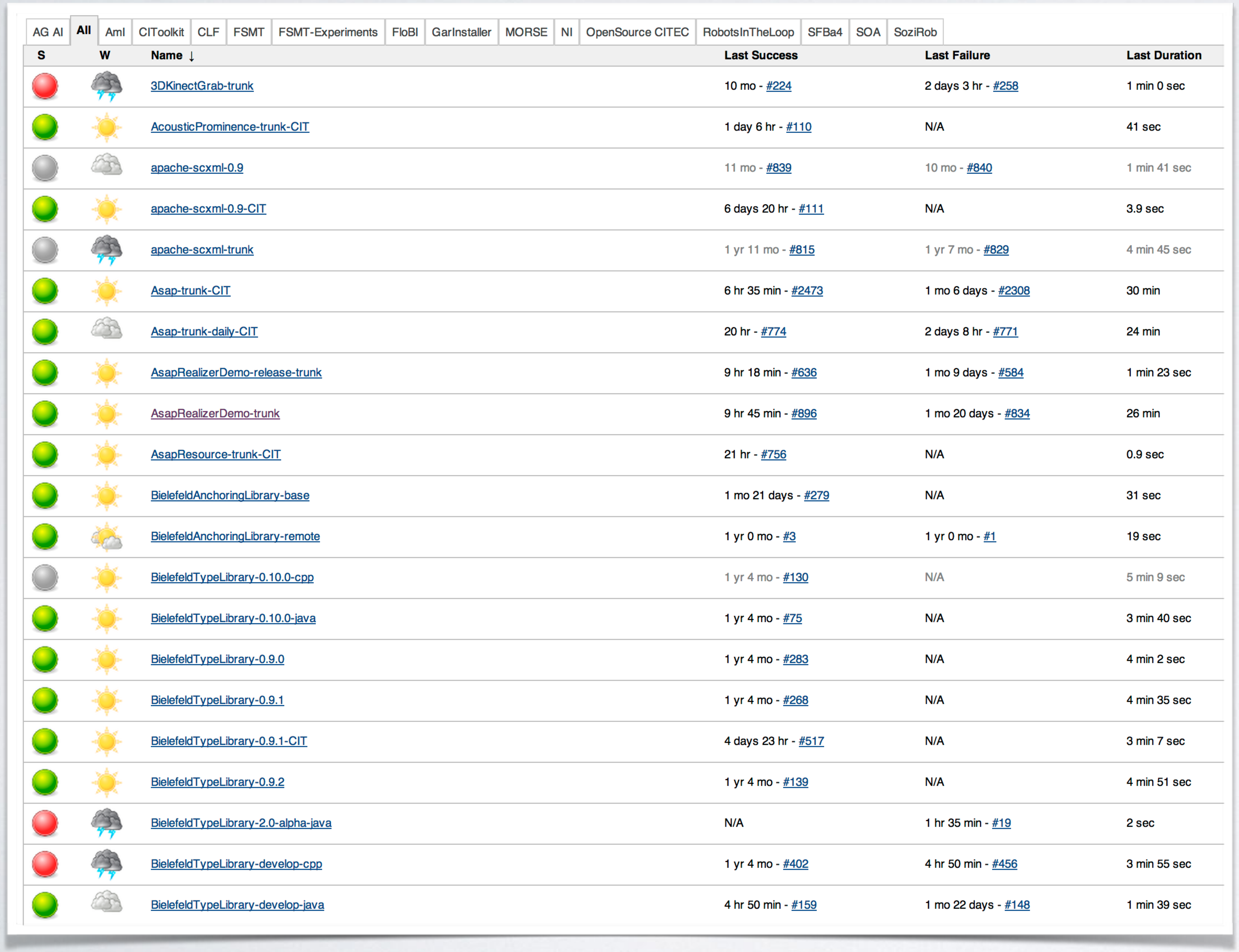
Later elaborations of the concept introduced build servers, which automatically run the unit tests periodically or even after every commit and report the results to the developers. The use of build servers (not necessarily running unit tests) had already been practised by some teams outside the XP community. Nowadays, many organisations have adopted CI without adopting all of XP.

In addition to automated unit tests, organisations using CI typically use a build server to implement *continuous* processes of applying quality control in general — small pieces of effort, applied frequently. In addition to running the unit and integration tests, such processes run additional static and dynamic tests, measure and profile performance, extract and format documentation from the source code and facilitate manual QA processes. This continuous application of quality control aims to improve the quality of software, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control *after* completing all development. This is very similar to the original idea of integrating more frequently to make integration easier, only applied to QA processes.

In the same vein the practice of continuous delivery further extends CI by making sure the software checked in on the mainline is always in a state that can be deployed to users and makes the actual deployment process very rapid.

http://en.wikipedia.org/wiki/Continuous_integration

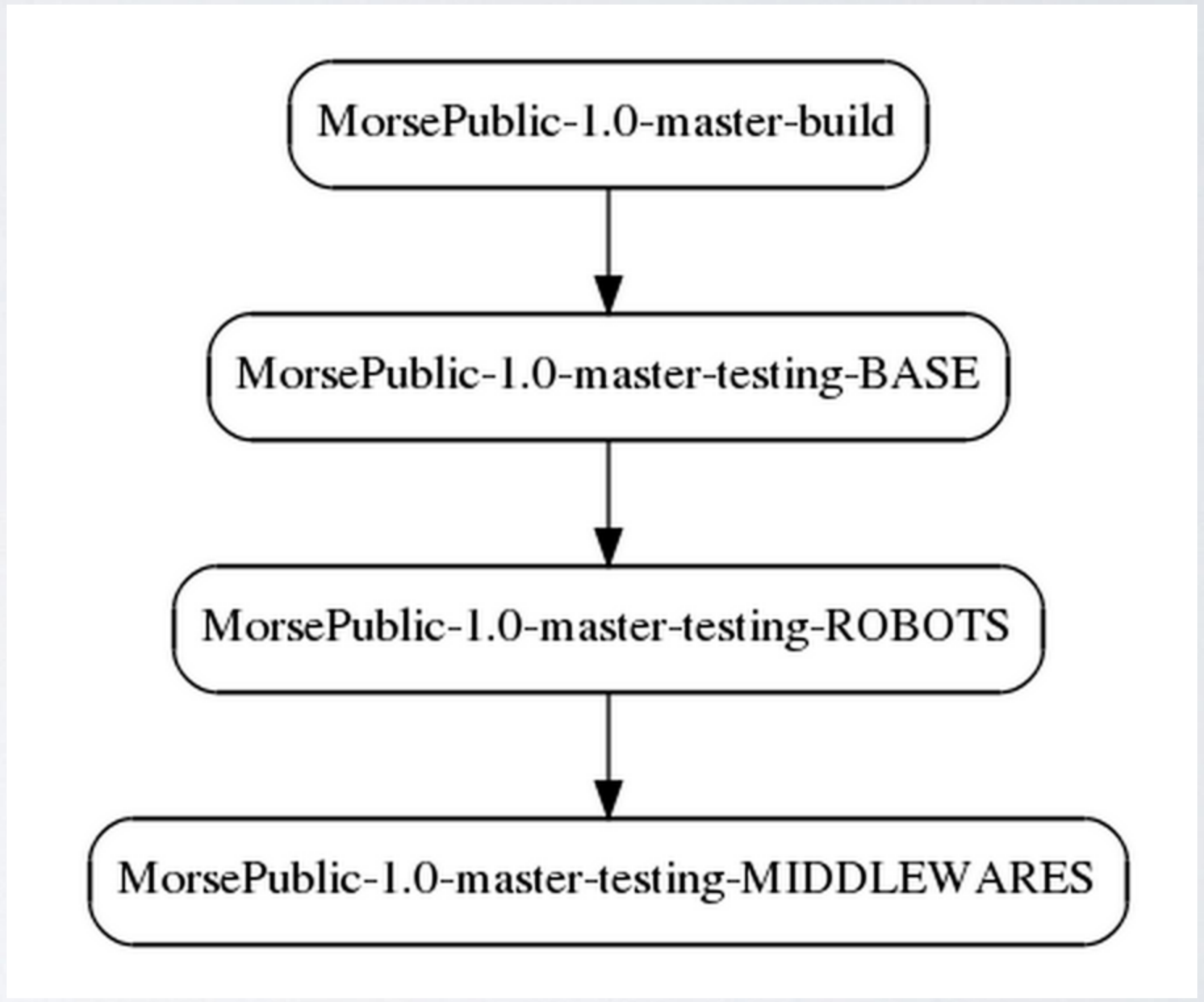
CI PRIMER



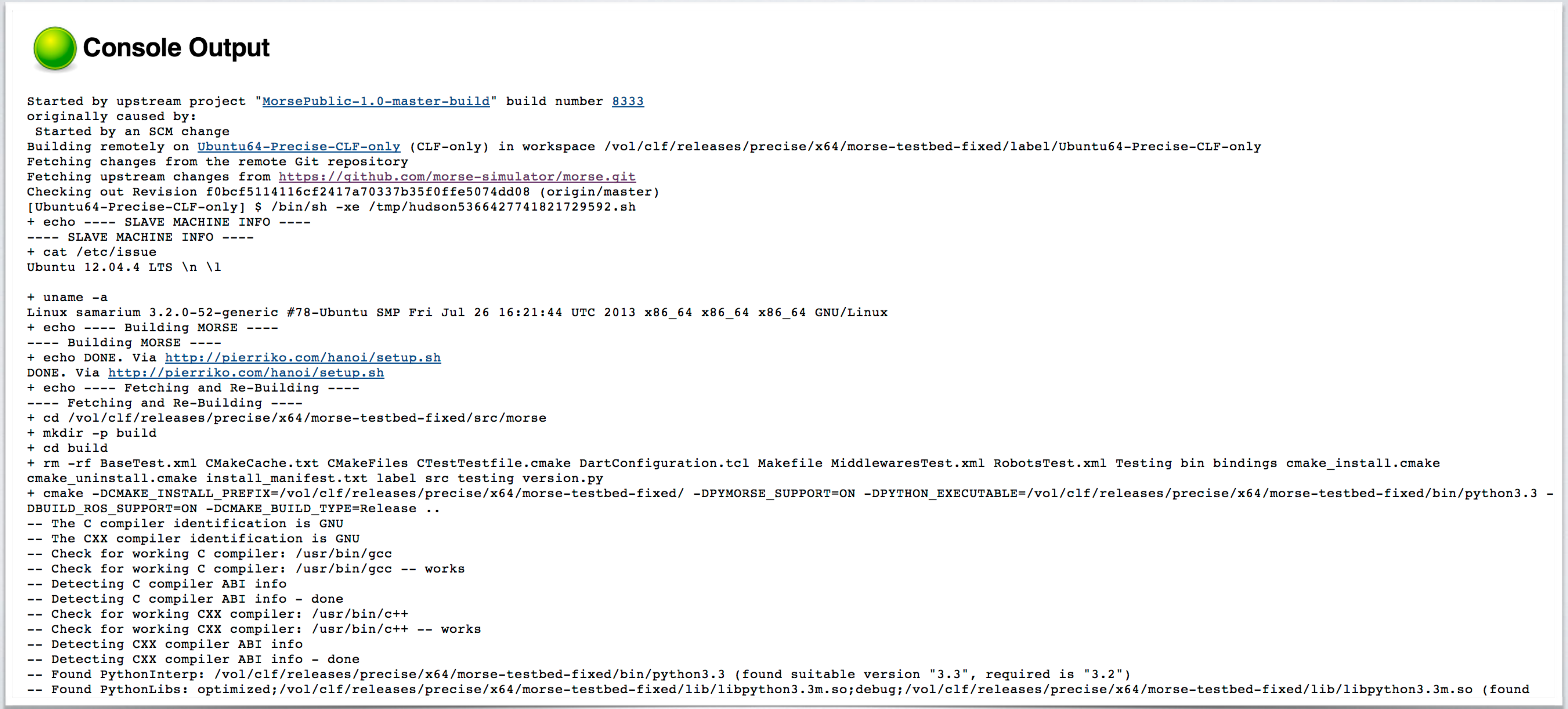
AG AI | **All** | AmI | CIToolkit | CLF | FSMT | FSMT-Experiments | FloBI | GarInstaller | MORSE | NI | OpenSource CITEC | RobotsInTheLoop | SFBa4 | SOA | SoziRob

S	W	Name ↓	Last Success	Last Failure	Last Duration
🔴	⛈	3DKinectGrab-trunk	10 mo - #224	2 days 3 hr - #258	1 min 0 sec
🟢	☀	AcousticProminence-trunk-CIT	1 day 6 hr - #110	N/A	41 sec
⚪	☁	apache-scxml-0.9	11 mo - #839	10 mo - #840	1 min 41 sec
🟢	☀	apache-scxml-0.9-CIT	6 days 20 hr - #111	N/A	3.9 sec
⚪	⛈	apache-scxml-trunk	1 yr 11 mo - #815	1 yr 7 mo - #829	4 min 45 sec
🟢	☀	Asap-trunk-CIT	6 hr 35 min - #2473	1 mo 6 days - #2308	30 min
🟢	☁	Asap-trunk-daily-CIT	20 hr - #774	2 days 8 hr - #771	24 min
🟢	☀	AsapRealizerDemo-release-trunk	9 hr 18 min - #636	1 mo 9 days - #584	1 min 23 sec
🟢	☀	AsapRealizerDemo-trunk	9 hr 45 min - #896	1 mo 20 days - #834	26 min
🟢	☀	AsapResource-trunk-CIT	21 hr - #756	N/A	0.9 sec
🟢	☀	BielefeldAnchoringLibrary-base	1 mo 21 days - #279	N/A	31 sec
🟢	⛅	BielefeldAnchoringLibrary-remote	1 yr 0 mo - #3	1 yr 0 mo - #1	19 sec
⚪	☀	BielefeldTypeLibrary-0.10.0-cpp	1 yr 4 mo - #130	N/A	5 min 9 sec
🟢	☀	BielefeldTypeLibrary-0.10.0-java	1 yr 4 mo - #75	N/A	3 min 40 sec
🟢	☀	BielefeldTypeLibrary-0.9.0	1 yr 4 mo - #283	N/A	4 min 2 sec
🟢	☀	BielefeldTypeLibrary-0.9.1	1 yr 4 mo - #268	N/A	4 min 35 sec
🟢	☀	BielefeldTypeLibrary-0.9.1-CIT	4 days 23 hr - #517	N/A	3 min 7 sec
🟢	☀	BielefeldTypeLibrary-0.9.2	1 yr 4 mo - #139	N/A	4 min 51 sec
🔴	⛈	BielefeldTypeLibrary-2.0-alpha-java	N/A	1 hr 35 min - #19	2 sec
🔴	⛈	BielefeldTypeLibrary-develop-cpp	1 yr 4 mo - #402	4 hr 50 min - #456	3 min 55 sec
🟢	☁	BielefeldTypeLibrary-develop-java	4 hr 50 min - #159	1 mo 22 days - #148	1 min 39 sec

http://jenkins-ci.org/


MorsePublic-1.0-master-build
MorsePublic-1.0-master-testing-BASE
MorsePublic-1.0-master-testing-ROBOTS
MorsePublic-1.0-master-testing-MIDDLEWARES

CI PRIMER

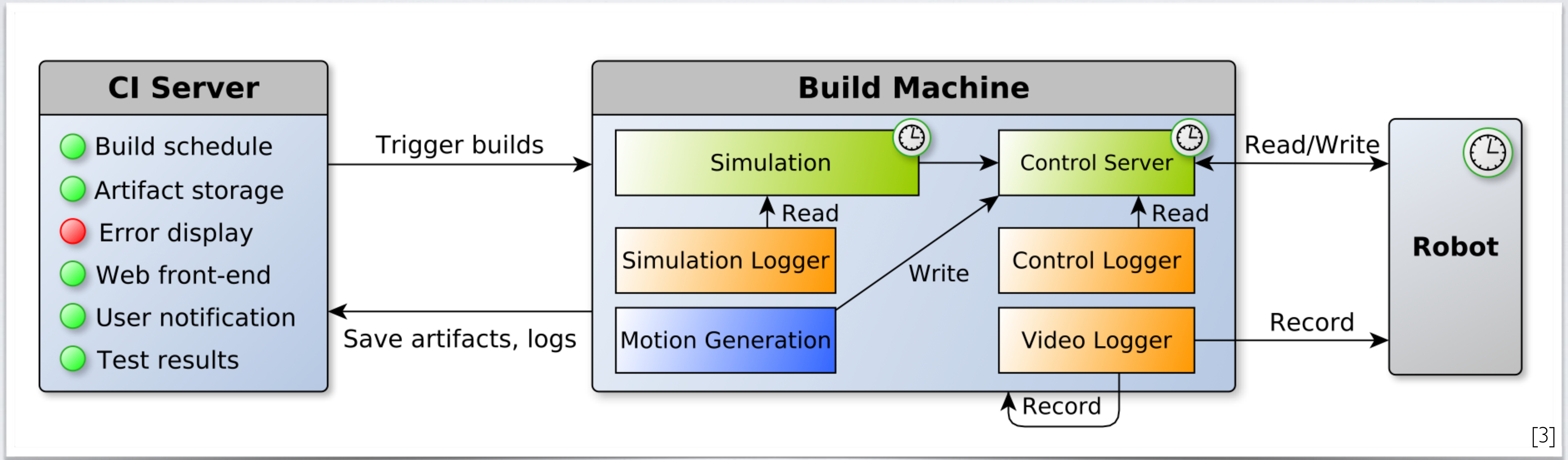


Console Output

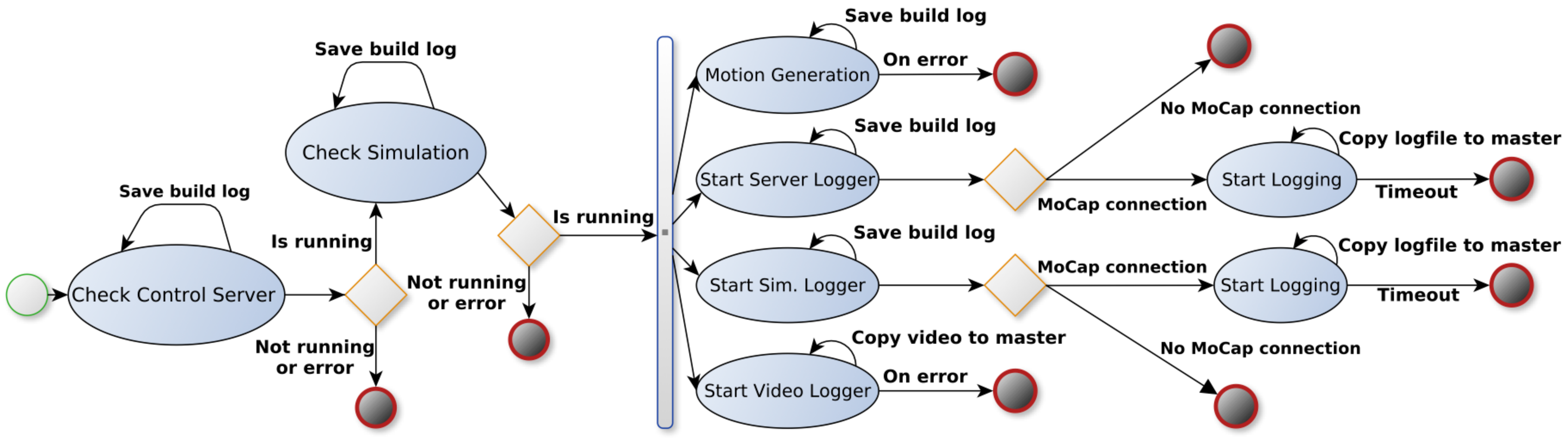
```
Started by upstream project "MorsePublic-1.0-master-build" build number 8333
originally caused by:
 Started by an SCM change
Building remotely on Ubuntu64-Precise-CLF-only (CLF-only) in workspace /vol/clf/releases/precise/x64/morse-testbed-fixed/label/Ubuntu64-Precise-CLF-only
Fetching changes from the remote Git repository
Fetching upstream changes from https://github.com/morse-simulator/morse.git
Checking out Revision f0bcf5114116cf2417a70337b35f0ffe5074dd08 (origin/master)
[Ubuntu64-Precise-CLF-only] $ /bin/sh -xe /tmp/hudson5366427741821729592.sh
+ echo ---- SLAVE MACHINE INFO ----
---- SLAVE MACHINE INFO ----
+ cat /etc/issue
Ubuntu 12.04.4 LTS \n \l

+ uname -a
Linux samarium 3.2.0-52-generic #78-Ubuntu SMP Fri Jul 26 16:21:44 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
+ echo ---- Building MORSE ----
---- Building MORSE ----
+ echo DONE. Via http://pierriko.com/hanoi/setup.sh
DONE. Via http://pierriko.com/hanoi/setup.sh
+ echo ---- Fetching and Re-Building ----
---- Fetching and Re-Building ----
+ cd /vol/clf/releases/precise/x64/morse-testbed-fixed/src/morse
+ mkdir -p build
+ cd build
+ rm -rf BaseTest.xml CMakeCache.txt CMakeFiles CTestTestfile.cmake DartConfiguration.tcl Makefile MiddlewaresTest.xml RobotsTest.xml Testing bin bindings cmake_install.cmake
cmake_uninstall.cmake install_manifest.txt label src testing version.py
+ cmake -DCMAKE_INSTALL_PREFIX=/vol/clf/releases/precise/x64/morse-testbed-fixed/ -DPYMORSE_SUPPORT=ON -DPYTHON_EXECUTABLE=/vol/clf/releases/precise/x64/morse-testbed-fixed/bin/python3.3 -
DBUILD_ROS_SUPPORT=ON -DCMAKE_BUILD_TYPE=Release ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Found PythonInterp: /vol/clf/releases/precise/x64/morse-testbed-fixed/bin/python3.3 (found suitable version "3.3", required is "3.2")
-- Found PythonLibs: optimized;/vol/clf/releases/precise/x64/morse-testbed-fixed/lib/libpython3.3m.so;debug;/vol/clf/releases/precise/x64/morse-testbed-fixed/lib/libpython3.3m.so (found
```

http://jenkins-ci.org/

FLOBI SIMULATION VALIDATION

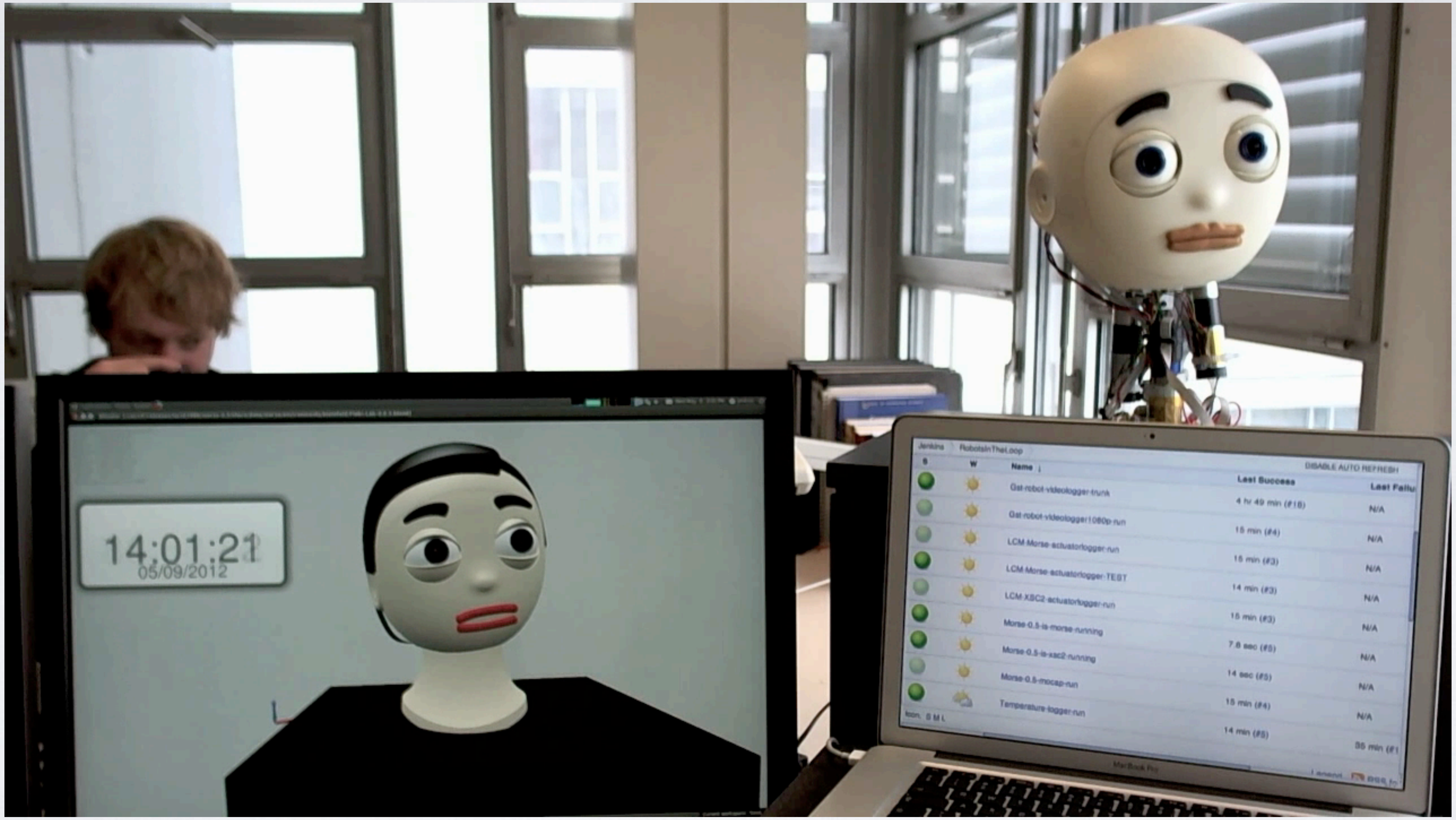


FLOBI SIMULATION VALIDATION



[3]

FLOBI SIMULATION VALIDATION



Jenkins › RobotsInTheLoop

DISABLE AUTO REFRESH

S	W	Name ↓	Last Success	Last Failu
●	☀	Gst-robot-videologger-trunk	4 hr 49 min (#18)	N/A
○	☀	Gst-robot-videologger1080p-run	15 min (#4)	N/A
○	☀	LCM-Morse-actuatorlogger-run	15 min (#3)	N/A
●	☀	LCM-Morse-actuatorlogger-TEST	14 min (#3)	N/A
○	☀	LCM-XSC2-actuatorlogger-run	15 min (#3)	N/A
●	☀	Morse-0.5-is-morse-running	7.8 sec (#5)	N/A
●	☀	Morse-0.5-is-xsc2-running	14 sec (#5)	N/A
○	☀	Morse-0.5-mocap-run	15 min (#4)	N/A
●	⛅	Temperature-logger-run	14 min (#5)	35 min (#1

Icon: S M L

[3]

FLOBI SIMULATION VALIDATION

Worked well, so far…

Does **not scale** for multiple projects/test

Time consuming configuration via GUI

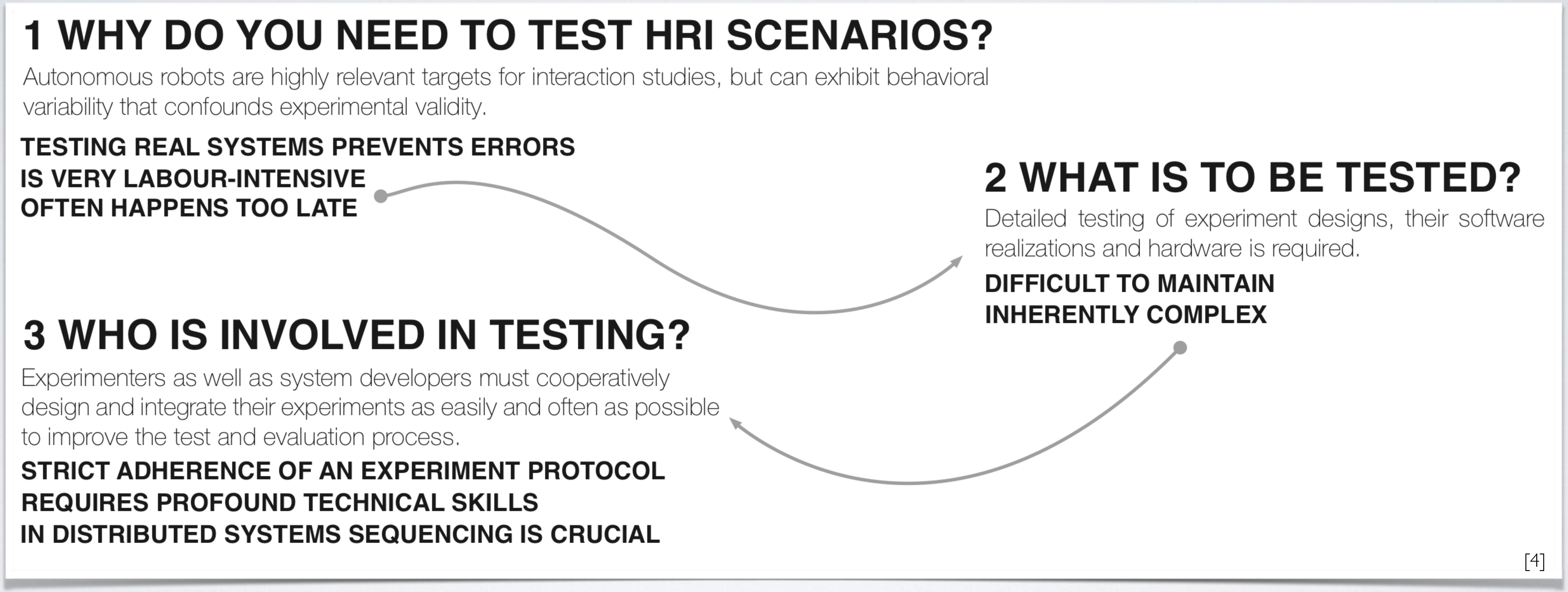
Not 100% **deterministic** (default Jenkins job scheduler)

Requires profound "Jenkins knowledge" to setup test scenarios

Technology for **developers**, which often excludes, i.e., social sciences

New approach: FSMT

FINITE-STATE-MACHINE-BASED-TESTING



1 WHY DO YOU NEED TO TEST HRI SCENARIOS?

Autonomous robots are highly relevant targets for interaction studies, but can exhibit behavioral variability that confounds experimental validity.

TESTING REAL SYSTEMS PREVENTS ERRORS
IS VERY LABOUR-INTENSIVE
OFTEN HAPPENS TOO LATE

2 WHAT IS TO BE TESTED?

Detailed testing of experiment designs, their software realizations and hardware is required.

DIFFICULT TO MAINTAIN
INHERENTLY COMPLEX

3 WHO IS INVOLVED IN TESTING?

Experimenters as well as system developers must cooperatively design and integrate their experiments as easily and often as possible to improve the test and evaluation process.

STRICT ADHERENCE OF AN EXPERIMENT PROTOCOL
REQUIRES PROFOUND TECHNICAL SKILLS
IN DISTRIBUTED SYSTEMS SEQUENCING IS CRUCIAL

[4]

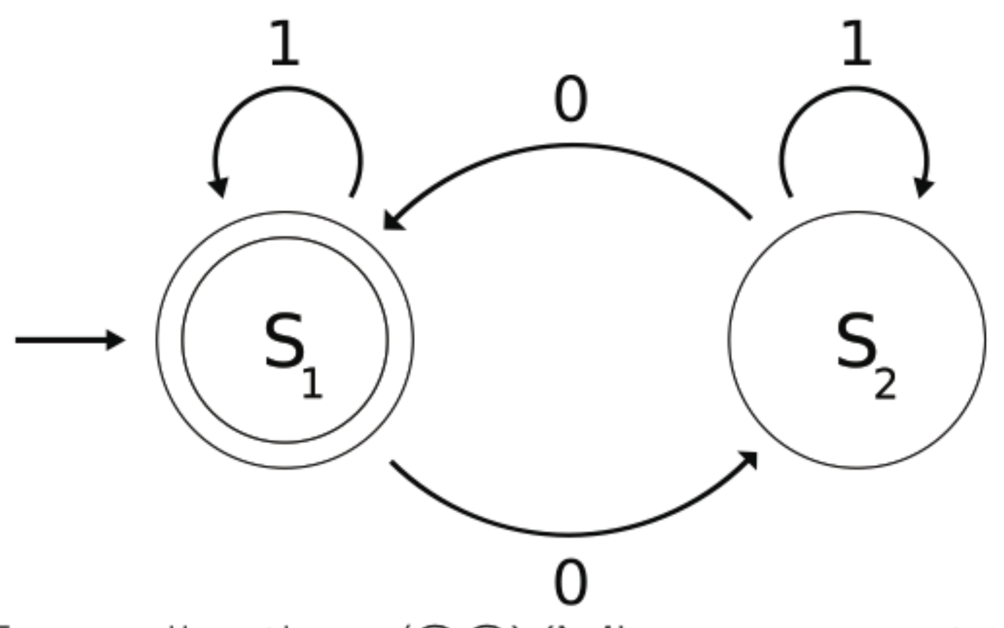
FINITE-STATE-MACHINE-BASED-TESTING

a) Establish experiment prototyping. This uses simulation environments including a virtual human component **b)** Extend the concept of an experiment protocol to the orchestration of software components, and **c)** Execute and assess the results of a prototype experiment in an automated, easy-to-use fashion.

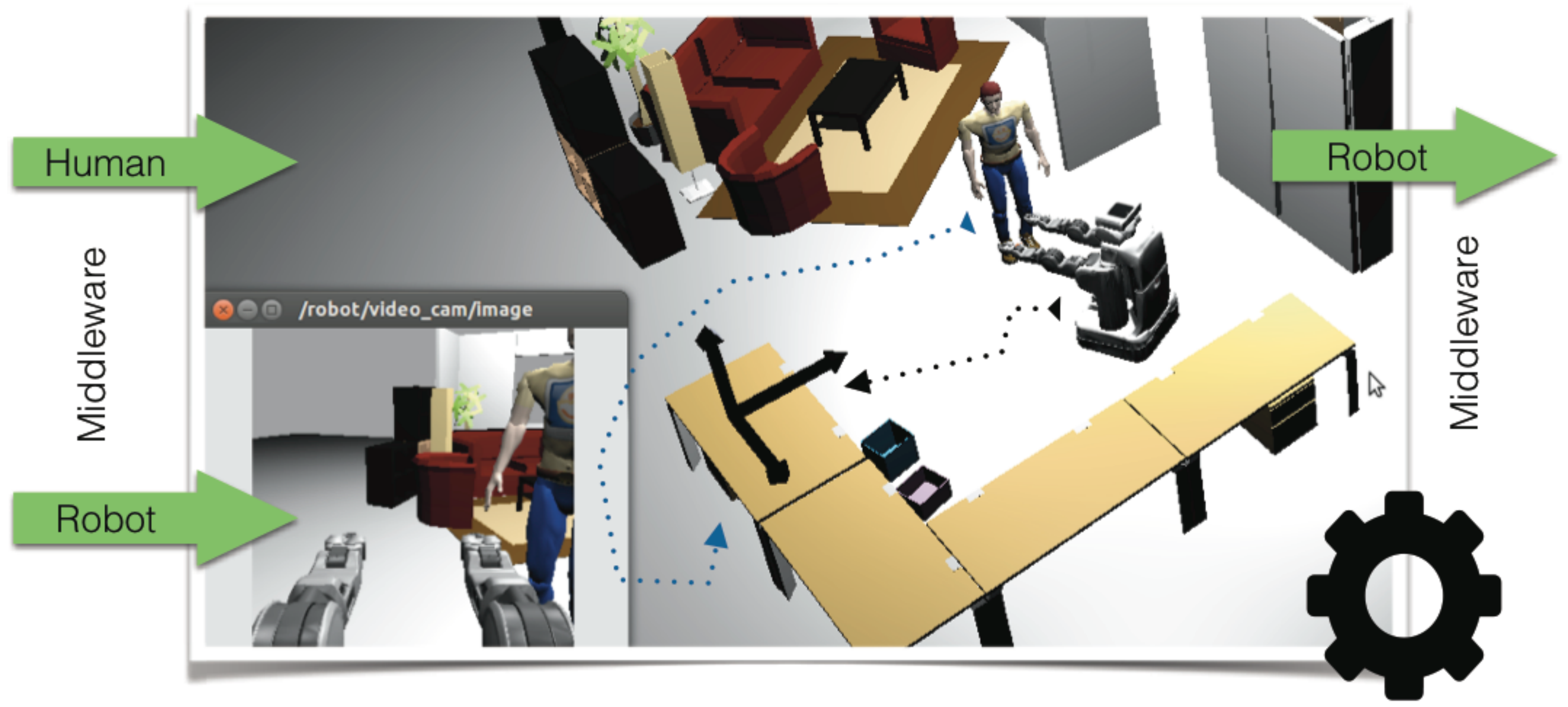
FSMT [2]

```
; Morse and ROS Testing Movement Assessment
[environment]
FSMPREFIX=/media/FSM-Lab/releases/precise/x64/
MORSE_ROOT=/media/FSM-Lab/releases/precise/x64/

[component-1]
name = eval_human_pose
command = python eval_xyz_pose.py
path = /opt/ros/groovy/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,8
blocking = True,True
ongoing = True,False
criteria = ,started core service

[component-2]
name = morse
command = morse run flobi_sim
path = $FSMPREFIX/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,5
blocking = True,True
ongoing = False,False
criteria = ,correctly setup to run MORSE

[run]
name = MORSE_Test
run_order = (‘morse‘,‘other_component‘)
run_execution_duration = 60
result_assessment_order = (‘eval_human_pose‘)
result_assessment_execution_duration = 10
```

Textual definition



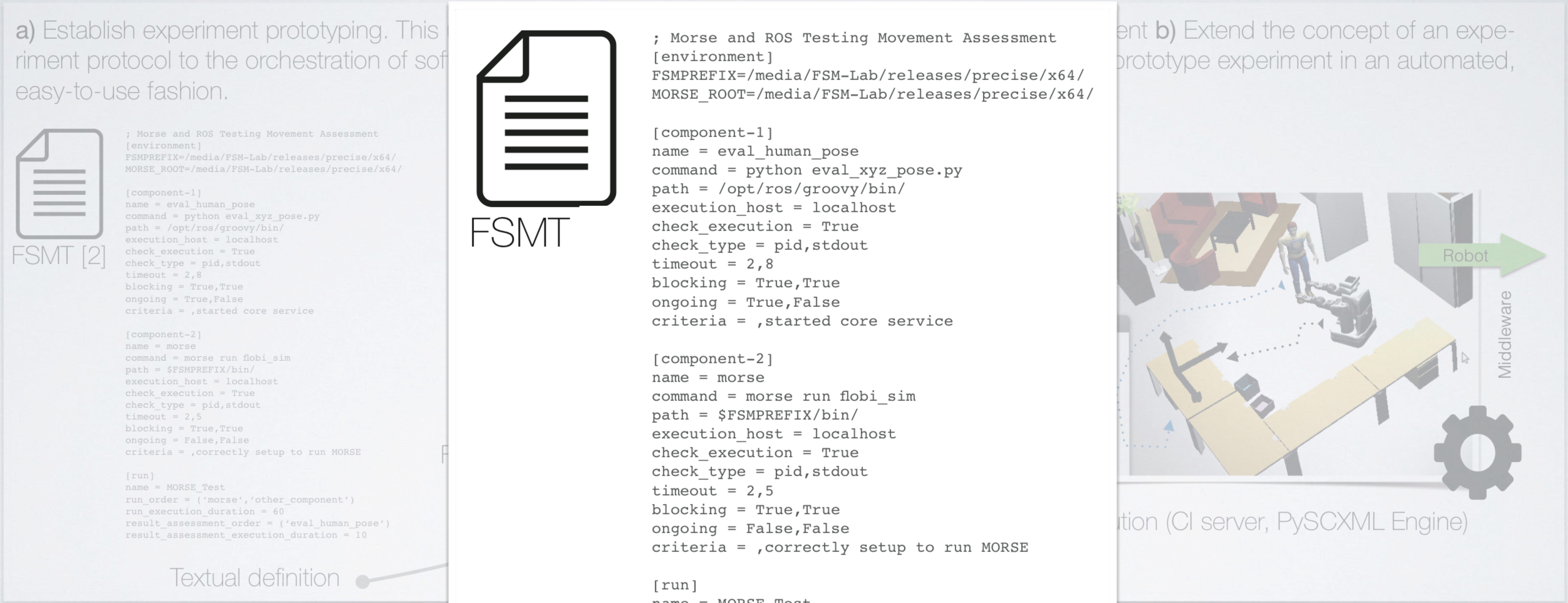
Formalization (SCXML representation)



Automated Execution (CI server, PySCXML Engine)

[4]

FINITE-STATE-MACHINE-BASED-TESTING


a) Establish experiment prototyping. This
riment protocol to the orchestration of sof
easy-to-use fashion.
FSMT [2]
Textual definition
ent b) Extend the concept of an expe-
prototype experiment in an automated,
Robot
Middleware
tion (CI server, PySCXML Engine)

```
; Morse and ROS Testing Movement Assessment
[environment]
FSMPREFIX=/media/FSM-Lab/releases/precise/x64/
MORSE_ROOT=/media/FSM-Lab/releases/precise/x64/

[component-1]
name = eval_human_pose
command = python eval_xyz_pose.py
path = /opt/ros/groovy/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,8
blocking = True,True
ongoing = True,False
criteria = ,started core service

[component-2]
name = morse
command = morse run flobi_sim
path = $FSMPREFIX/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,5
blocking = True,True
ongoing = False,False
criteria = ,correctly setup to run MORSE

[run]
name = MORSE_Test
run_order = ('morse','other_component')
run_execution_duration = 60
result_assessment_order = ('eval_human_pose')
result_assessment_execution_duration = 10
```

FSMT

```
; Morse and ROS Testing Movement Assessment
[environment]
FSMPREFIX=/media/FSM-Lab/releases/precise/x64/
MORSE_ROOT=/media/FSM-Lab/releases/precise/x64/

[component-1]
name = eval_human_pose
command = python eval_xyz_pose.py
path = /opt/ros/groovy/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,8
blocking = True,True
ongoing = True,False
criteria = ,started core service

[component-2]
name = morse
command = morse run flobi_sim
path = $FSMPREFIX/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,5
blocking = True,True
ongoing = False,False
criteria = ,correctly setup to run MORSE

[run]
name = MORSE_Test
run_order = ('morse','other_component')
run_execution_duration = 60
result_assessment_order = ('eval_human_pose')
result_assessment_execution_duration = 10
```

[4]

FINITE-STATE-MACHINE-BASED-TESTING

a) Establish experiment prototyping. This uses simulation environments including a virtual human component **b)** Extend the concept of an experiment protocol to the orchestration of software components, and **c)** Execute and assess the results of a prototype experiment in an automated, easy-to-use fashion.

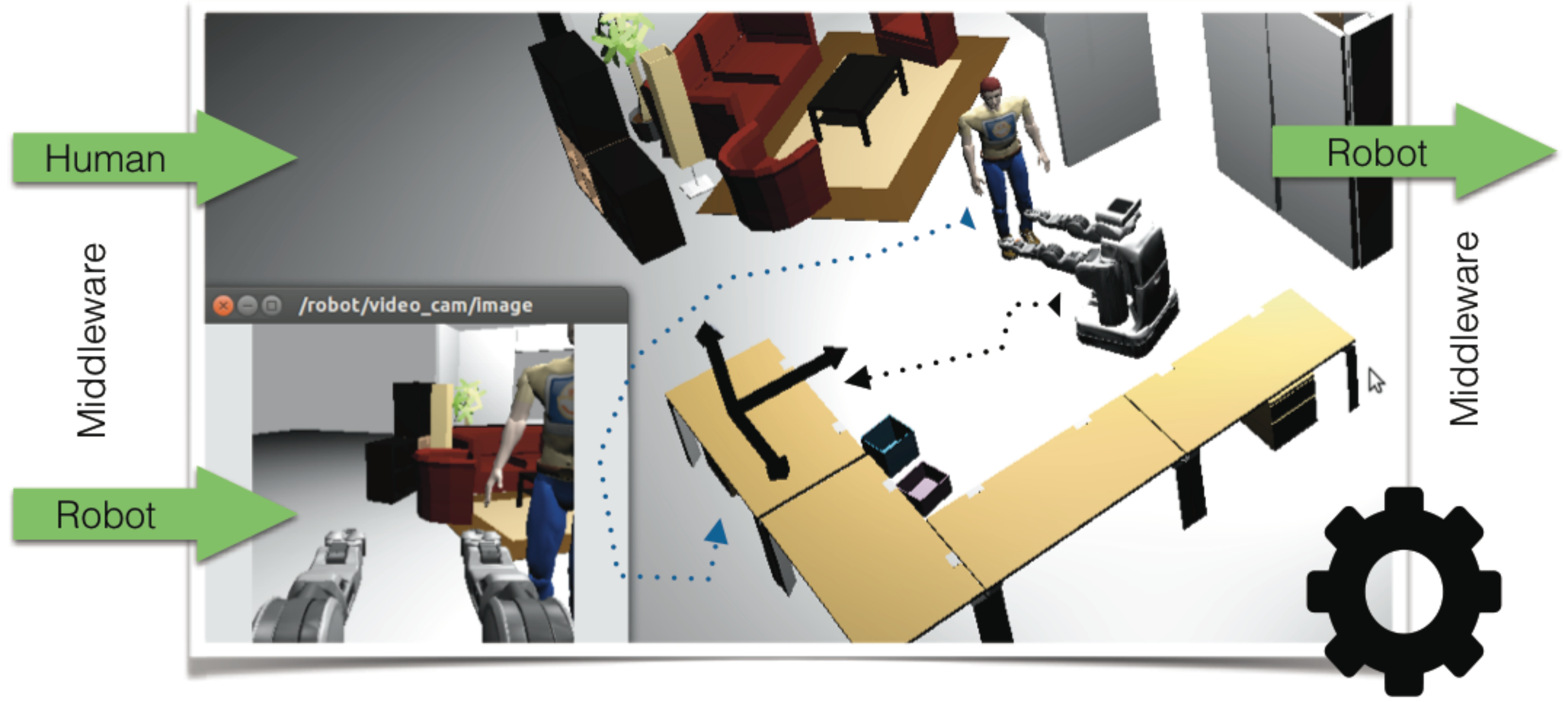
FSMT [2]

```
; Morse and ROS Testing Movement Assessment
[environment]
FSMPREFIX=/media/FSM-Lab/releases/precise/x64/
MORSE_ROOT=/media/FSM-Lab/releases/precise/x64/

[component-1]
name = eval_human_pose
command = python eval_xyz_pose.py
path = /opt/ros/groovy/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,8
blocking = True,True
ongoing = True,False
criteria = ,started core service

[component-2]
name = morse
command = morse run flobi_sim
path = $FSMPREFIX/bin/
execution_host = localhost
check_execution = True
check_type = pid,stdout
timeout = 2,5
blocking = True,True
ongoing = False,False
criteria = ,correctly setup to run MORSE

[run]
name = MORSE_Test
run_order = ('morse','other_component')
run_execution_duration = 60
result_assessment_order = ('eval_human_pose')
result_assessment_execution_duration = 10
```

Textual definition

$$\begin{tikzcd} {} \arrow[r] & S_1 \arrow[loop above, "1"] \arrow[rr, bend right, "0"'] & & S_2 \arrow[loop above, "1"] \arrow[ll, bend right, "0"'] \end{tikzcd}$$

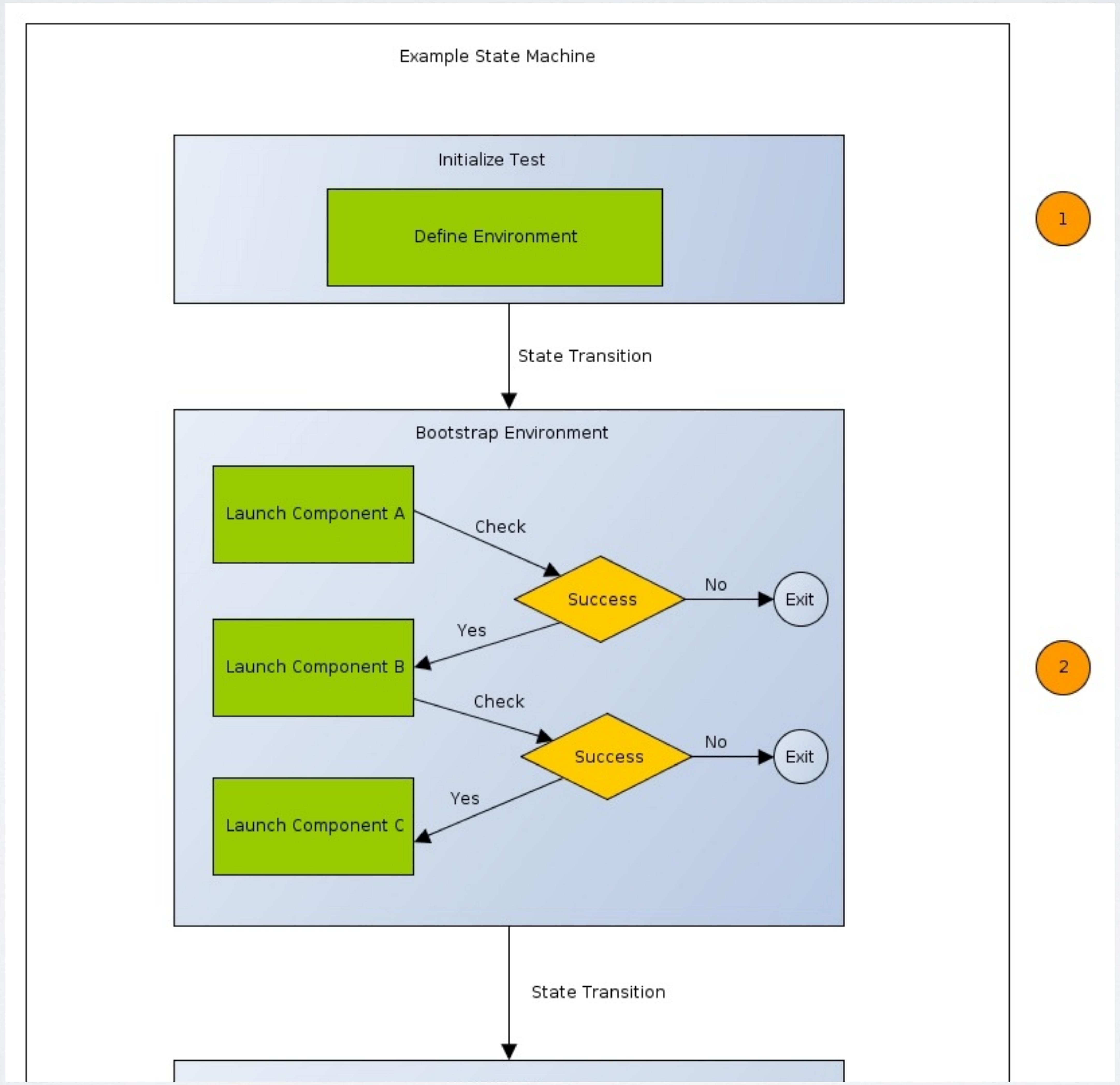
Formalization (SCXML representation)



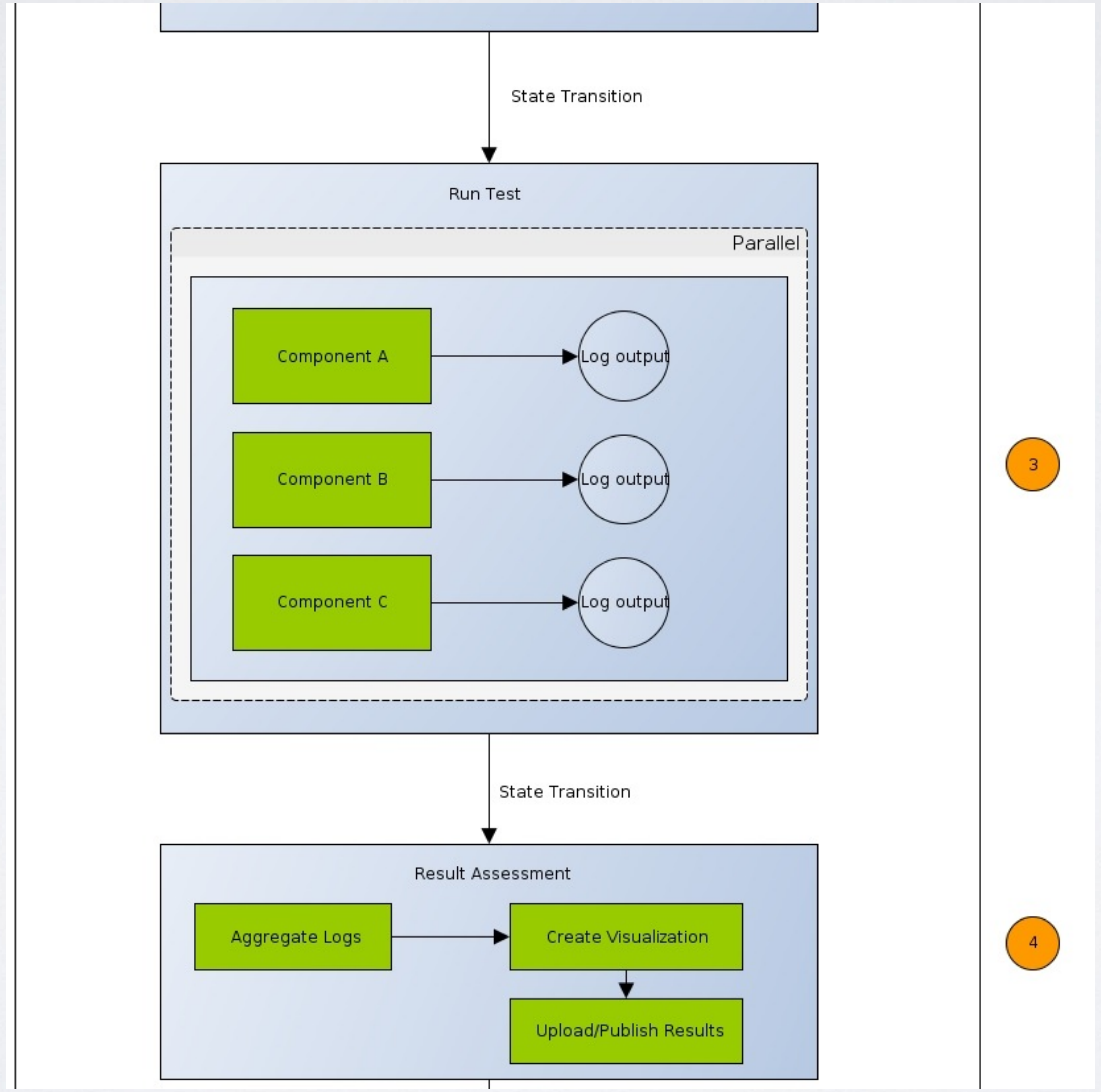
Automated Execution (CI server, PySCXML Engine)

[4]

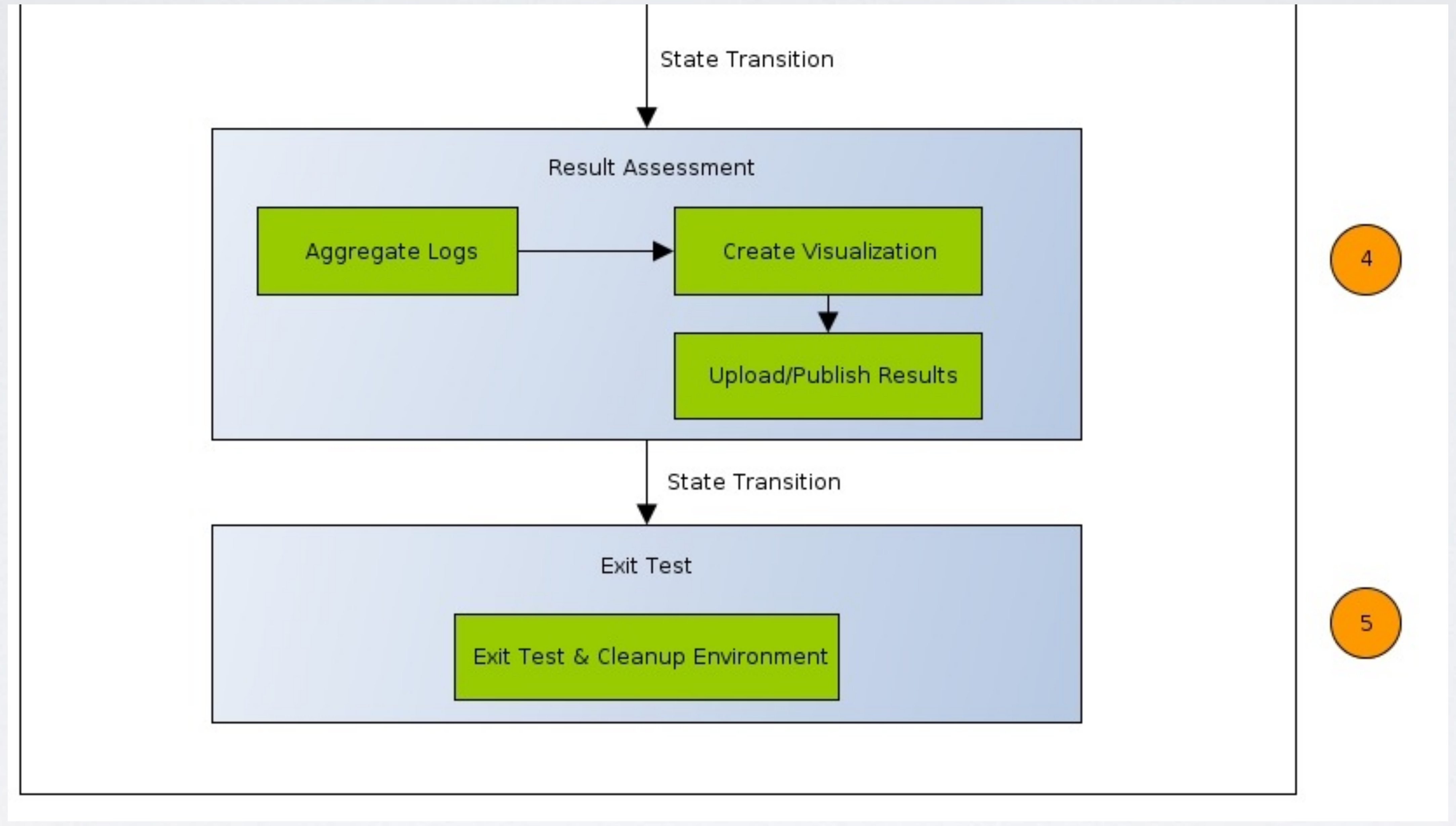
FINITE-STATE-MACHINE-BASED-TESTING



FINITE-STATE-MACHINE-BASED-TESTING

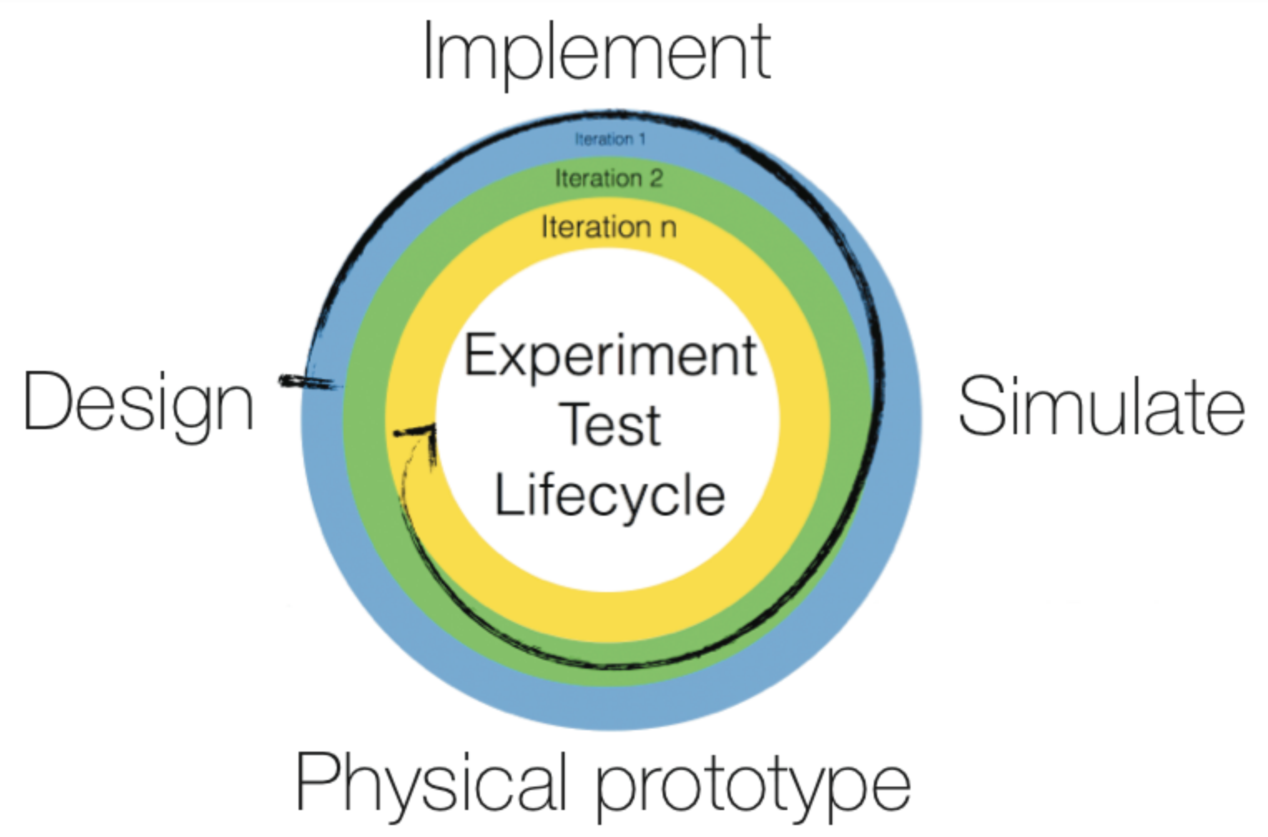


FINITE-STATE-MACHINE-BASED-TESTING



FINITE-STATE-MACHINE-BASED-TESTING

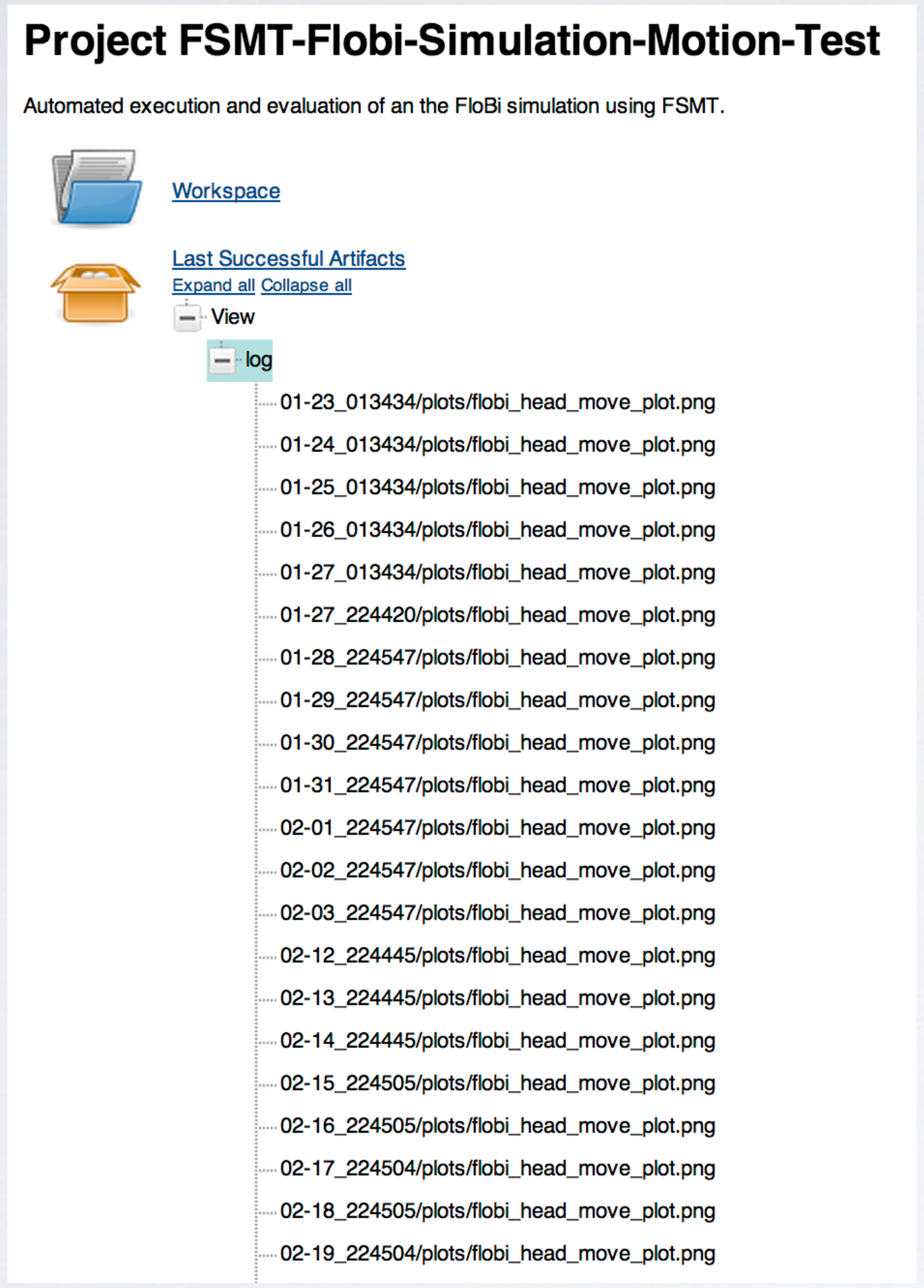
Provided benefits
No Framework “lock-in”
Human readable *.ini syntax
Automated translation into a SM
Already includes evaluation scripts



Provided benefits
Archiving of results
Early and continous feedback
Centralized testing infrastructure
Gradually increase level of simulation complexity

[4]

FINITE-STATE-MACHINE-BASED-TESTING



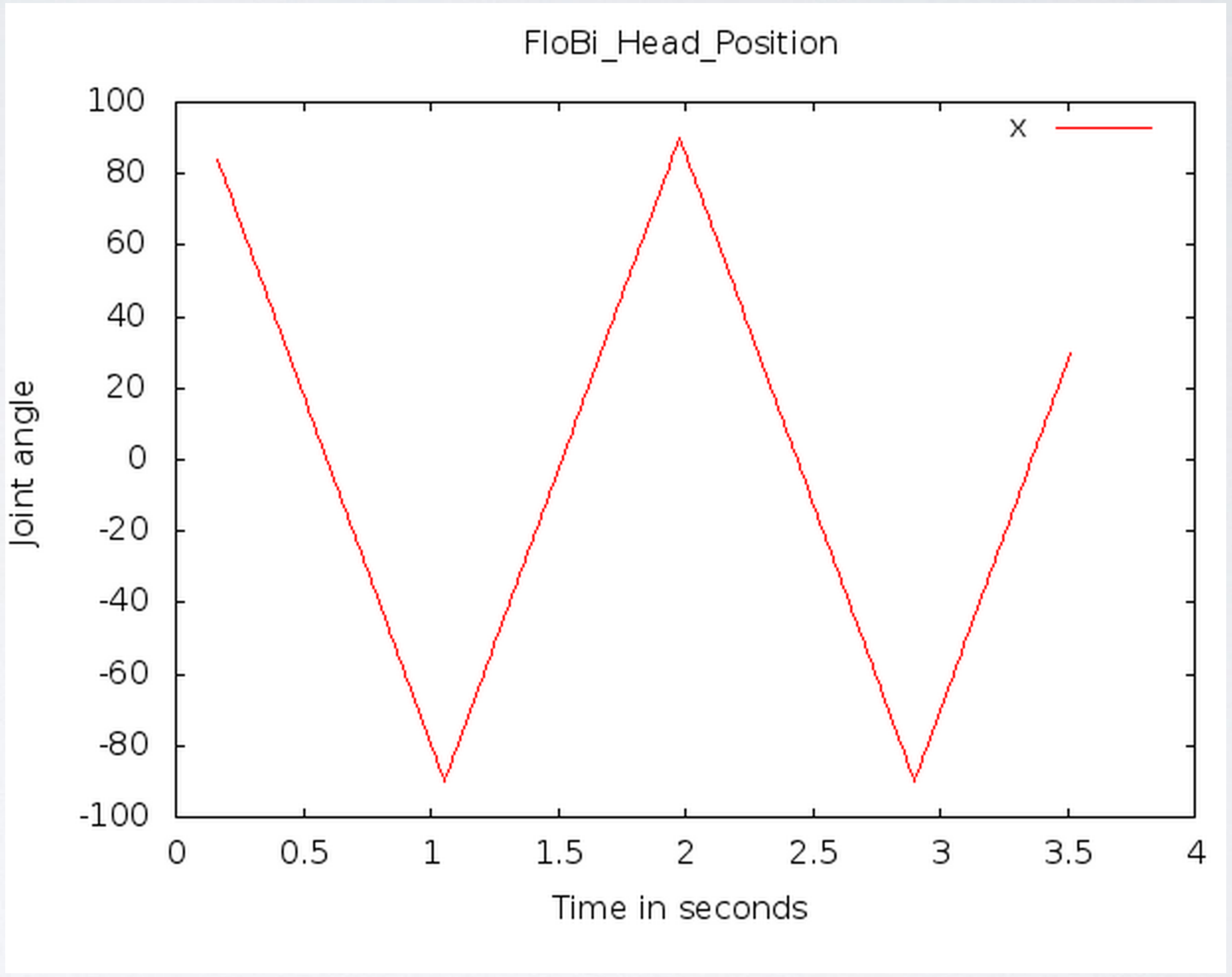
Project FSMT-Flobi-Simulation-Motion-Test

Automated execution and evaluation of an the FloBi simulation using FSMT.

Workspace

Last Successful Artifacts
Expand all Collapse all

- View
  - log
    - 01-23_013434/plots/flobi_head_move_plot.png
    - 01-24_013434/plots/flobi_head_move_plot.png
    - 01-25_013434/plots/flobi_head_move_plot.png
    - 01-26_013434/plots/flobi_head_move_plot.png
    - 01-27_013434/plots/flobi_head_move_plot.png
    - 01-27_224420/plots/flobi_head_move_plot.png
    - 01-28_224547/plots/flobi_head_move_plot.png
    - 01-29_224547/plots/flobi_head_move_plot.png
    - 01-30_224547/plots/flobi_head_move_plot.png
    - 01-31_224547/plots/flobi_head_move_plot.png
    - 02-01_224547/plots/flobi_head_move_plot.png
    - 02-02_224547/plots/flobi_head_move_plot.png
    - 02-03_224547/plots/flobi_head_move_plot.png
    - 02-12_224445/plots/flobi_head_move_plot.png
    - 02-13_224445/plots/flobi_head_move_plot.png
    - 02-14_224445/plots/flobi_head_move_plot.png
    - 02-15_224505/plots/flobi_head_move_plot.png
    - 02-16_224505/plots/flobi_head_move_plot.png
    - 02-17_224504/plots/flobi_head_move_plot.png
    - 02-18_224505/plots/flobi_head_move_plot.png
    - 02-19_224504/plots/flobi_head_move_plot.png



LIVE Demo :)

Of course we will make FSMT open soon

https://opensource.cit-ec.de

Thank you!

REFERENCES

[1] Wachsmuth S, Schulz S, Lier F, Lütkebohle I. The Robot Head "Flobi": A Research Platform for Cognitive Interaction Technology. Presented at the German Conference on Artificial Intelligence, Saarbrücken

[2] Wienke J, Wrede S. A Middleware for Collaborative Research in Experimental Robotics. In: IEEE/SICE International Symposium on System Integration (SII2011). IEEE; 2011: 1183–1190.

[3] Lier F, Lütkebohle I. Continuous Integration for Iterative Validation of Simulated Robot Models. In: SIMPAR 2012 proceedings. Lecture Notes in Computer Science, 7628. Berlin: Springer-Verlag; 2012.

[4] Lier F, Lütkebohle I, Wachsmuth S. Towards Automated Execution and Evaluation of Simulated Prototype HRI Experiments. In: Human-Robot Interaction (HRI), 2014 9th ACM/IEEE. Bielefeld, Germany: Human-Robot Interaction (HRI), 2014 9th ACM/IEEE;

[5] Echeverria, Gilberto, et al. "Modular open robots simulation engine: Morse." Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011.