WEPKS005

ICALEPCS'11

13th International Conference on Accelerator and Large Experimental Physics Control Systems
October 10-14, 2011, WTC Grenoble, France

State Machine Framework and its Use for Driving LHC Operational States

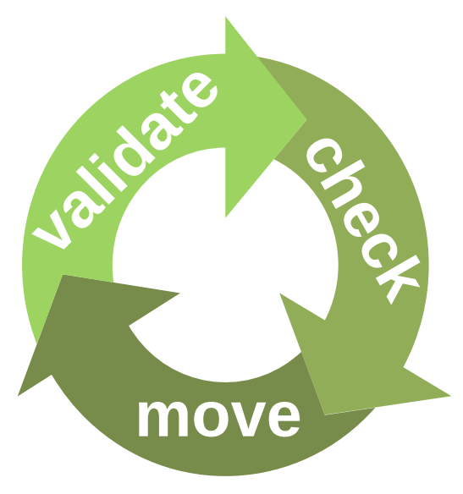
M. Misiowiec, V. Baggiolini, M. Solfaroli Camillocci, CERN, Geneva, Switzerland

LHC follows a complex operational cycle with 12 major phases that include equipment tests, preparation, beam injection, ramping and squeezing, finally followed by the physics phase. This cycle is modelled and enforced with a state machine, whereby each operational phase is represented by a state. On each transition, before entering the next state, a series of conditions is verified to make sure the LHC is ready to move on. Java State Machine Framework was developed to cater for building independent or embedded state machines. They safely drive between the states executing tasks bound to transitions and broadcast related information to clients. SM framework encourages users to create their own actions. Simple configuration management allows the operators to define and maintain complex models themselves.

State Machine Framework

State Machine is composed of ***states*** connected by ***transitions*** and associated with ***actions***. Exactly one state is ***initial***. State machine holds a status with an ***active*** state as its most essential attribute.

Move



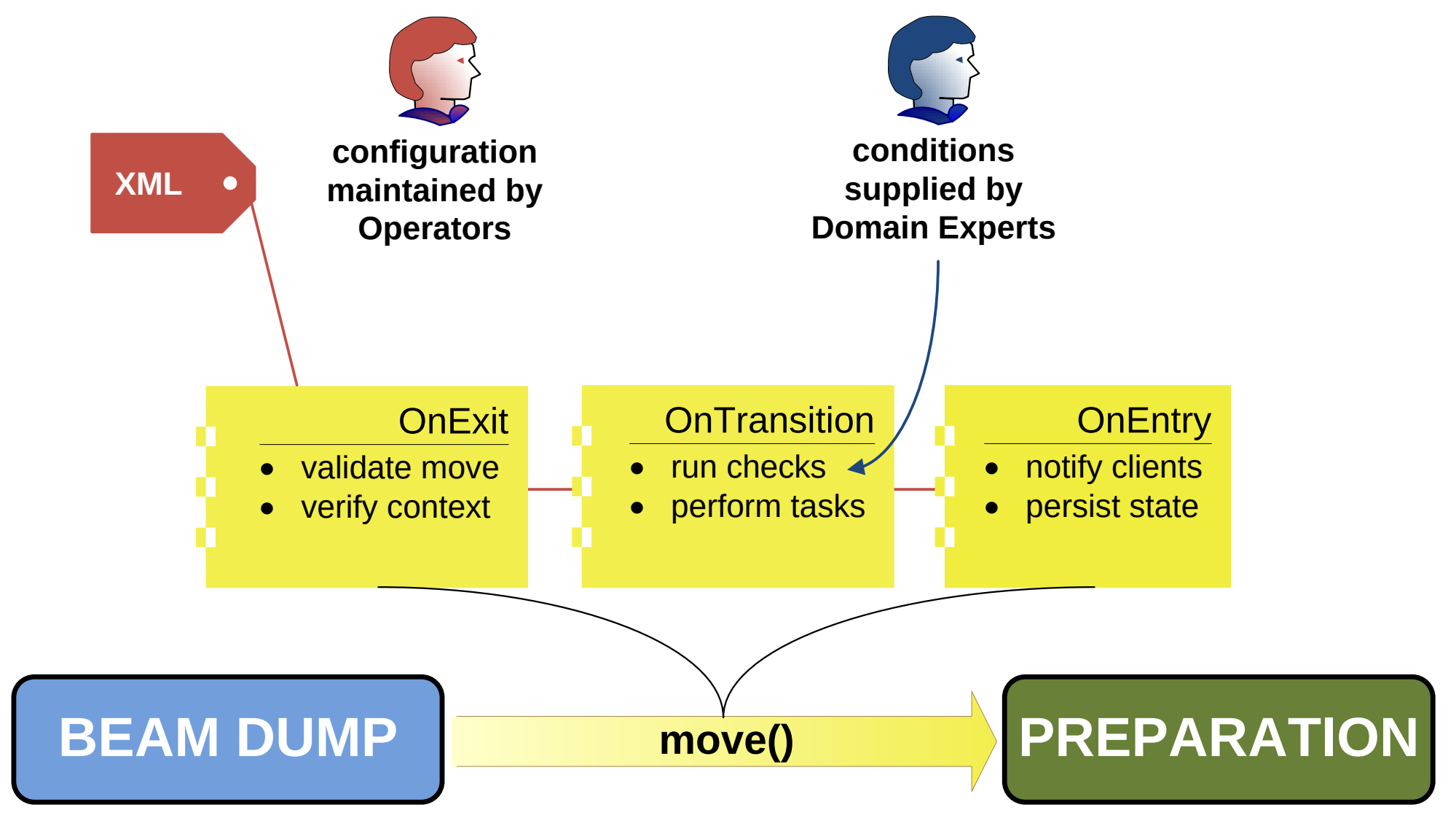
Moving from state A to B is ***valid*** only if A is active and there exists a transition from A to B. Set of actions can be assigned to the model and required to be successfully executed during transitions.

Actions Development

Actions are:

- ***tasks*** – operations executed in the environment
- ***conditions*** – checks performed on the environment

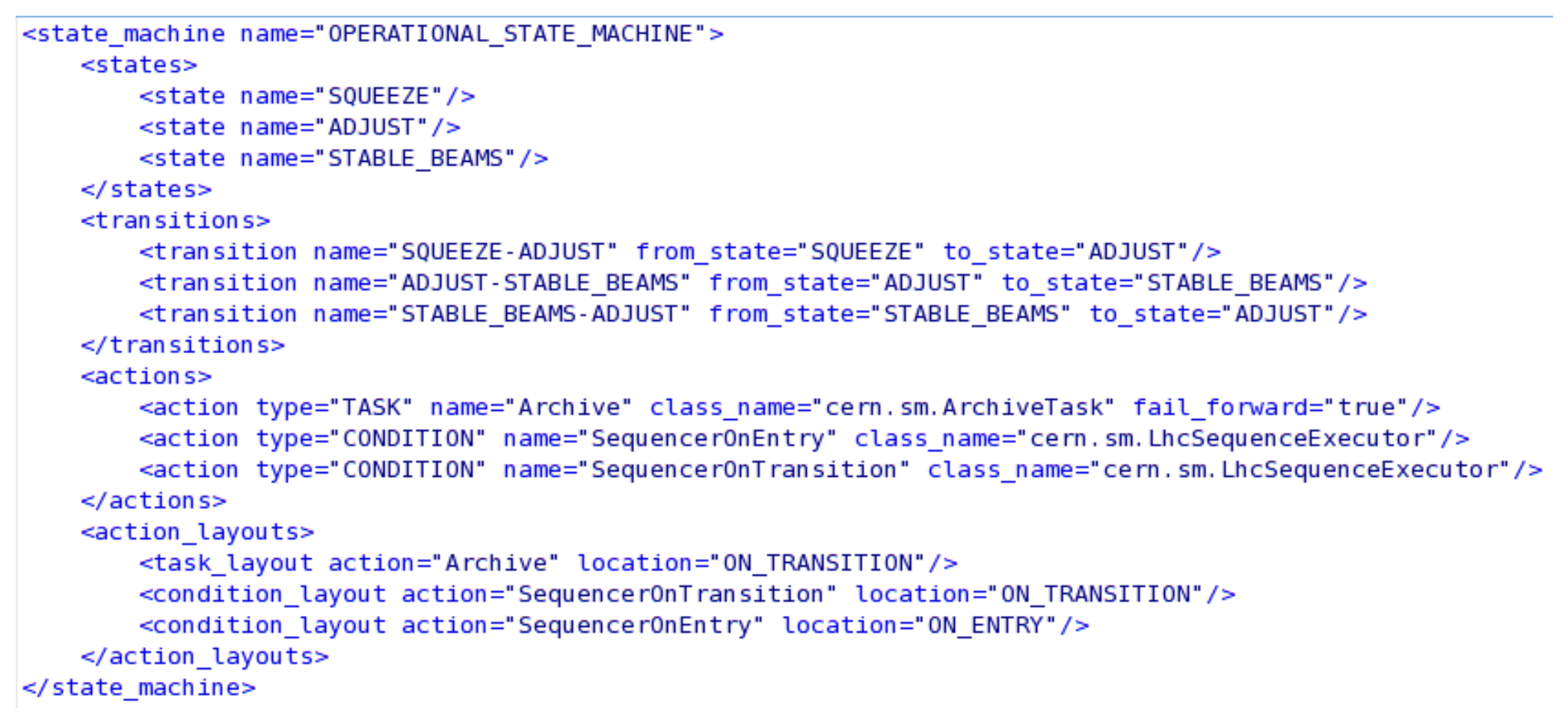
Actions are developed to the dedicated API by ***domain experts*** and associated by configuration. Variety of information is provided to the action execution, e.g. client credentials. An action can be placed in one of three locations with regard to a transition or state: OnExit, OnTransition or OnEntry.



OnExit and OnEntry actions are associated with states and executed for all the transitions leaving or reaching a state. OnTransition actions are linked to a transition.

Configuration

SMF provides a high degree of flexibility in configuring the layout of transitions and actions. Configuration of SM instance is held in XML or database. In each case, template schema is provided to verify the syntax.

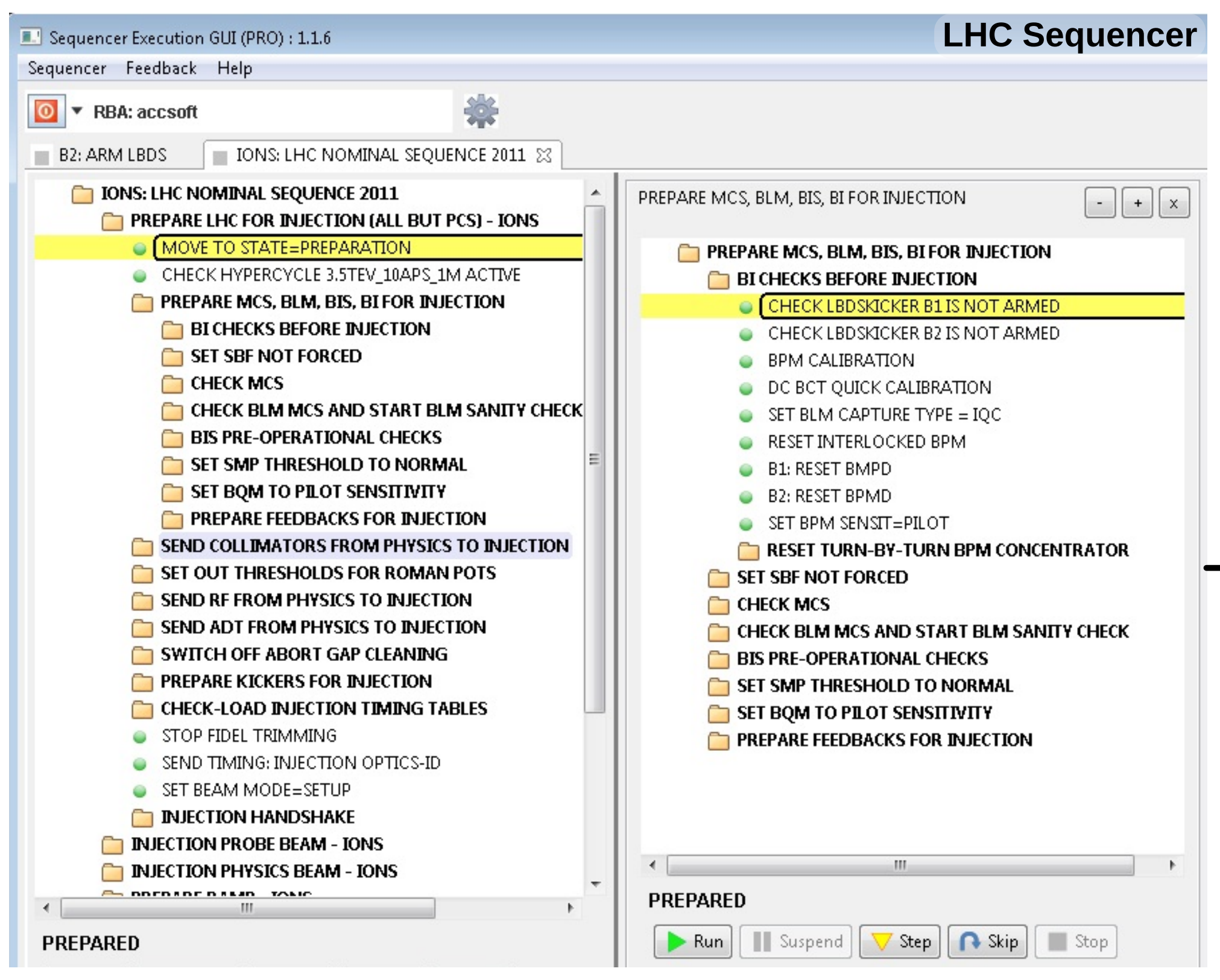


```
<state_machine name="OPERATIONAL_STATE_MACHINE">
    <states>
        <state name="SQUEEZE"/>
        <state name="ADJUST"/>
        <state name="STABLE_BEAMS"/>
    </states>
    <transitions>
        <transition name="SQUEEZE-ADJUST" from_state="SQUEEZE" to_state="ADJUST"/>
        <transition name="ADJUST-STABLE_BEAMS" from_state="ADJUST" to_state="STABLE_BEAMS"/>
        <transition name="STABLE_BEAMS-ADJUST" from_state="STABLE_BEAMS" to_state="ADJUST"/>
    </transitions>
    <actions>
        <action type="TASK" name="Archive" class_name="cern.sm.ArchiveTask" fail_forward="true"/>
        <action type="CONDITION" name="SequencerOnEntry" class_name="cern.sm.LhcSequenceExecutor"/>
        <action type="CONDITION" name="SequencerOnTransition" class_name="cern.sm.LhcSequenceExecutor"/>
    </actions>
    <action_layouts>
        <task_layout action="Archive" location="ON_TRANSITION"/>
        <condition_layout action="SequencerOnTransition" location="ON_TRANSITION"/>
        <condition_layout action="SequencerOnEntry" location="ON_ENTRY"/>
    </action_layouts>
</state_machine>
```

XML document, readable and simple to maintain, was the choice for LHC Operational States. It has let **operators manage** the configuration fully on their own which has proven stimulating for maintenance of the instance. XML configuration is stored in SVN under version control.

Deployment in LHC

LHC Sequencer executes preconfigured series of tasks – **sequences**. They move the machine through its operational lifecycle. Each phase of the lifecycle, an LHC operational state, is guarded by a set of conditions – **checks** – supervised by **LHC Operational States** state machine service.



LHC Sequencer

Sequencer Execution GUI (PRO) : 1.1.6
Sequencer Feedback Help
RBA: accsoft
B2: ARM LBDS | IONS: LHC NOMINAL SEQUENCE 2011

IONS: LHC NOMINAL SEQUENCE 2011
  PREPARE LHC FOR INJECTION (ALL BUT PCS) - IONS
    MOVE TO STATE=PREPARATION
    CHECK HYPERCYCLE 3.5TEV_10APS_1M ACTIVE
    PREPARE MCS, BLM, BIS, BI FOR INJECTION
      BI CHECKS BEFORE INJECTION
      SET SBF NOT FORCED
      CHECK MCS
      CHECK BLM MCS AND START BLM SANITY CHECK
      BIS PRE-OPERATIONAL CHECKS
      SET SMP THRESHOLD TO NORMAL
      SET BQM TO PILOT SENSITIVITY
      PREPARE FEEDBACKS FOR INJECTION
    SEND COLLIMATORS FROM PHYSICS TO INJECTION
    SET OUT THRESHOLDS FOR ROMAN POTS
    SEND RF FROM PHYSICS TO INJECTION
    SEND ADT FROM PHYSICS TO INJECTION
    SWITCH OFF ABORT GAP CLEANING
    PREPARE KICKERS FOR INJECTION
    CHECK-LOAD INJECTION TIMING TABLES
    STOP FIDEL TRIMMING
    SEND TIMING: INJECTION OPTICS-ID
    SET BEAM MODE=SETUP
    INJECTION HANDSHAKE
  INJECTION PROBE BEAM - IONS
  INJECTION PHYSICS BEAM - IONS

PREPARED

PREPARE MCS, BLM, BIS, BI FOR INJECTION

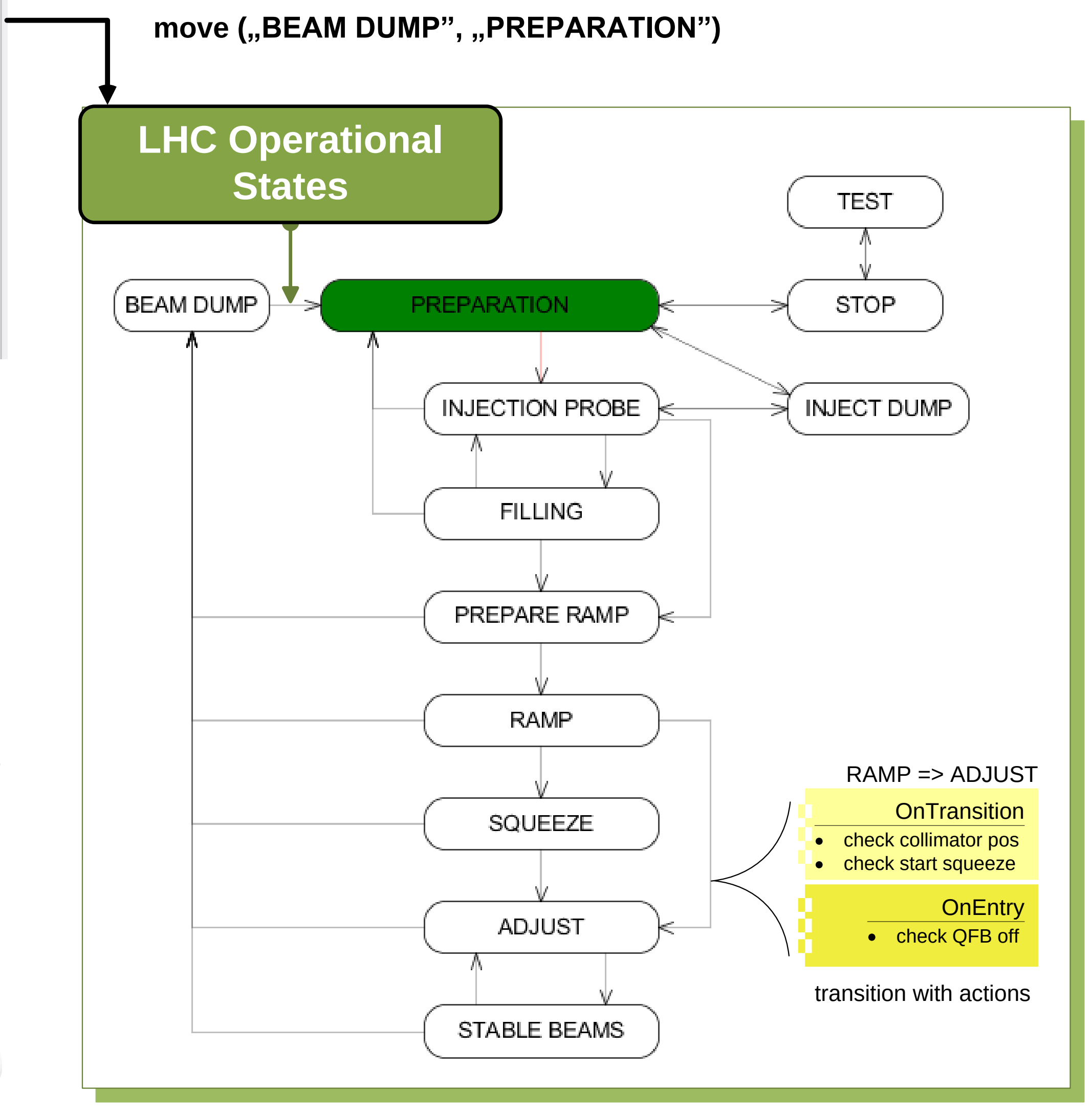
PREPARE MCS, BLM, BIS, BI FOR INJECTION
  BI CHECKS BEFORE INJECTION
    CHECK LBDSKICKER B1 IS NOT ARMED
    CHECK LBDSKICKER B2 IS NOT ARMED
    BPM CALIBRATION
    DC BCT QUICK CALIBRATION
    SET BLM CAPTURE TYPE = IQC
    RESET INTERLOCKED BPM
    B1: RESET BMPD
    B2: RESET BPMD
    SET BPM SENSIT=PILOT
    RESET TURN-BY-TURN BPM CONCENTRATOR
  SET SBF NOT FORCED
  CHECK MCS
  CHECK BLM MCS AND START BLM SANITY CHECK
  BIS PRE-OPERATIONAL CHECKS
  SET SMP THRESHOLD TO NORMAL
  SET BQM TO PILOT SENSITIVITY
  PREPARE FEEDBACKS FOR INJECTION

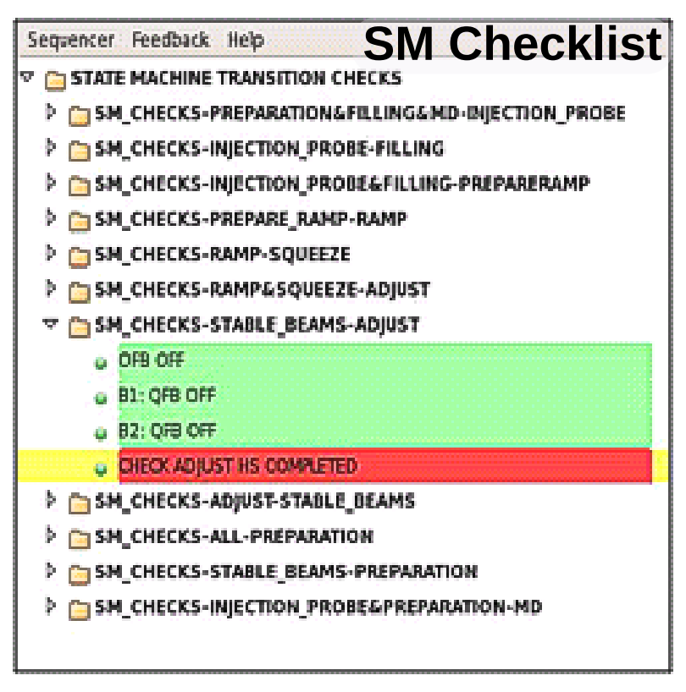
PREPARED

Run | Suspend | Step | Skip | Stop

LHC Operational States state machine instance manages transitions between the LHC states and monitors execution of the checks.

Client requests are limited with Role Based Access system to the operators of CERN Control Centre.





SM Checklist

Sequencer Feedback Help
STATE MACHINE TRANSITION CHECKS
  SM_CHECKS-PREPARATION&FILLING&MD-INJECTION_PROBE
  SM_CHECKS-INJECTION_PROBE-FILLING
  SM_CHECKS-INJECTION_PROBE&FILLING-PREPARERAMP
  SM_CHECKS-PREPARE_RAMP-RAMP
  SM_CHECKS-RAMP-SQUEEZE
  SM_CHECKS-RAMP&SQUEEZE-ADJUST
  SM_CHECKS-STABLE_BEAMS-ADJUST
    QFB OFF
    B1: QFB OFF
    B2: QFB OFF
    CHECK ADJUST HS COMPLETED
  SM_CHECKS-ADJUST-STABLE_BEAMS
  SM_CHECKS-ALL-PREPARATION
  SM_CHECKS-STABLE_BEAMS-PREPARATION
  SM_CHECKS-INJECTION_PROBE&PREPARATION-MD

SM GUIs

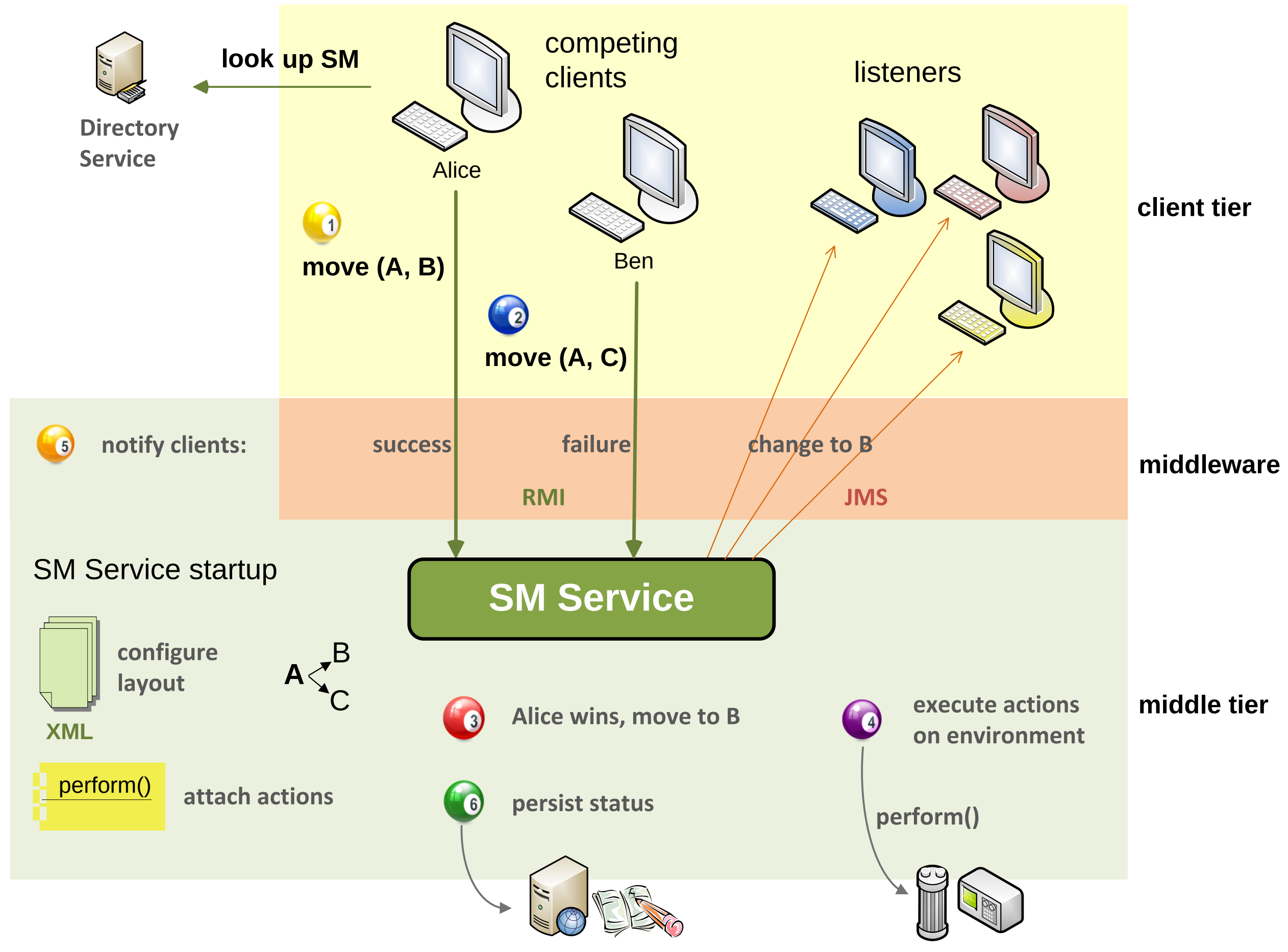
GUIs were developed to monitor the Sequencer requests and visualize the state machine model. Especially practical in the early stage of model design, they are used on a daily basis by LHC operators.

Completing Move

Results of the move are returned to the caller and asynchronously published to all the listening clients. They are also persisted and logged.

Designed for Quality

Muti-tiered Architecture



Safety

State machines open to a parallel use of many clients are prone to errors. Diligent effort was taken to design flawless concurrency engine serving multiple clients. A year of intensive use has confirmed a bug-free environment.

Embedding

Framework supports an embedded use within another application. The instance is hooked into the controlling application using Spring context. All the client calls are executed locally, between Java threads. Embedded mode is achieved by configuration.

Interoperability

Various middleware protocols are supported to communicate status messages from the SM instance.

Conclusions

State Machine Framework and its first instance have been operationally used for almost a year now. Over the period it has become an integral constituent of the LHC controls software architecture, running impeccably throughout. Its open architecture promoted active participation of the users, LHC operators, in the development and maintenance of both, the instance and framework.

State Machine Framework is a general purpose library aimed at both standalone or embedded use wherever state machine concepts need putting in place. Minimal dependency on the accelerator controls infrastructure makes it a comfortable choice for any project seeking a similar tool.

Perfect for:

- building a state machine service in a multi-tier system
- embedding state machine within any application
- outsourcing actions development
- easy maintenance of XML/DB configuration
- heavily concurrent environment