ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA187

ENHANCING THE MxCuBE USER INTERFACE BY A FINITE STATE MACHINE (FSM) MODEL

Ivars Karpics, Gleb Bourenkov, Thomas Schneider,
European Molecular Biology Laboratory (EMBL) Hamburg Unit
also at DESY, Notkestrasse 85, 22607, Hamburg, Germany

Abstract

The acquisition of X-ray diffraction data from macromolecular crystals is a major activity at many synchrotrons and requires user interfaces that provide robust and easy-to-use control of the experimental setup. Building on the modular design of the MxCuBE beamline user interface, we have implemented a finite state machine model that allows to describe and monitor the interaction of the user with the beamline in a typical experiment. Using a finite state machine, the path of user interaction can be rationalized and error conditions and recovery procedures can be systematically dealt with.

INTRODUCTION

EMBL Hamburg operates two beamlines, P13 and P14, for macromolecular X-ray crystallography (MX) on the PETRA III synchrotron at DESY (Hamburg, Germany). On both beamlines, MxCuBE [1] is used as the user interface. While in MX, a large fraction of data collections are considered as 'measurements' and are often conducted by non-experts, still many data collections can be considered as 'experiments' in which a fine control of data collection parameters is required to obtain usable data. Both aspects require the user interface to be robust and easy-to-use for measurements while providing flexibility and deeper control for experiments. To make the control of the beamlines more robust and intuitive from the user perspective, we have embarked on describing the interaction of the user with the beamline as a finite state machine (FSM). An FSM is a mathematical model of a closed or open loop discrete-event system with defined states [2]. FSM-graphs are widely used to define, analyse and control the functioning of system. Applications include software engineering and experimental control systems. For example, state graphs are used in Large Hadron Collider experiments at CERN [3, 4] and in data acquisition system at the European X-ray free-electron laser: Karabo SCADA framework [5]. For the case of controlling an experiment/measurement on an MX beamline, the modularity and clean separation between low-level hardware access and the graphical interface within MxCuBE allows implementing an FSM model into the main experimental cycle. Important features of using an FSM are the ability to have an overview of all relevant beamline components, to keep track of user actions, and to guide users and/or beamline staff in case of failure or alarm.

USE CASE

A highly-simplified typical user interaction with a macromolecular crystallography (MX) beamline for the collection of diffraction data includes mounting the sample onto a sample positioning device (goniometer), centering of the sample with respect to the X-ray beam, setting of the data collection parameters, triggering of the data collection, execution of the data collection which consists of rotating the sample in the X-ray beam, and finally unmounting of the sample. The MxCuBE graphical user interface (GUI) for MX beamlines contains numerous widgets to control the settings of the beamline hardware including X-ray energy, beam attenuation factors, size and shape of the X-ray beam, distance between the sample and X-ray detector, and others (Fig. 1). To facilitate sample centering, the user interface offers a life view of the sample and various means to position and orient the sample. The data collection parameters are set by entering numerical values for controlling sample rotation speeds, exposure times etc. The many different ways of collecting data on a given crystals and the interdependencies between different - often technically high-end and thus not ultimately stable - components of the beamline result in a highly complex system for which stable operation is not trivial to achieve. An important factor in this context is also the fact that many users of MX beamlines are inexperienced posing high requirements in terms of making the operation robust and supporting recovery from errors.

FINITE STATE MACHINE

In the case of FSM, a system can be in exactly one finite state. It can move to another state in response to defined external inputs (stimuli). An FSM is defined as a set of states and transitions, an initial state, and conditions for transitions. For a typical user interaction with the experimental GUI during an MX data collection, it is possible to indicate several common steps (transitions in the FSM graph):

1. Sample mounting on the sample positioning device (goniometer).
2. Sample centring.
3. Entry and validation of data collection parameters.
4. Execution of data collection.
5. Sample dismounting.
6. Returning to step 1.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA187

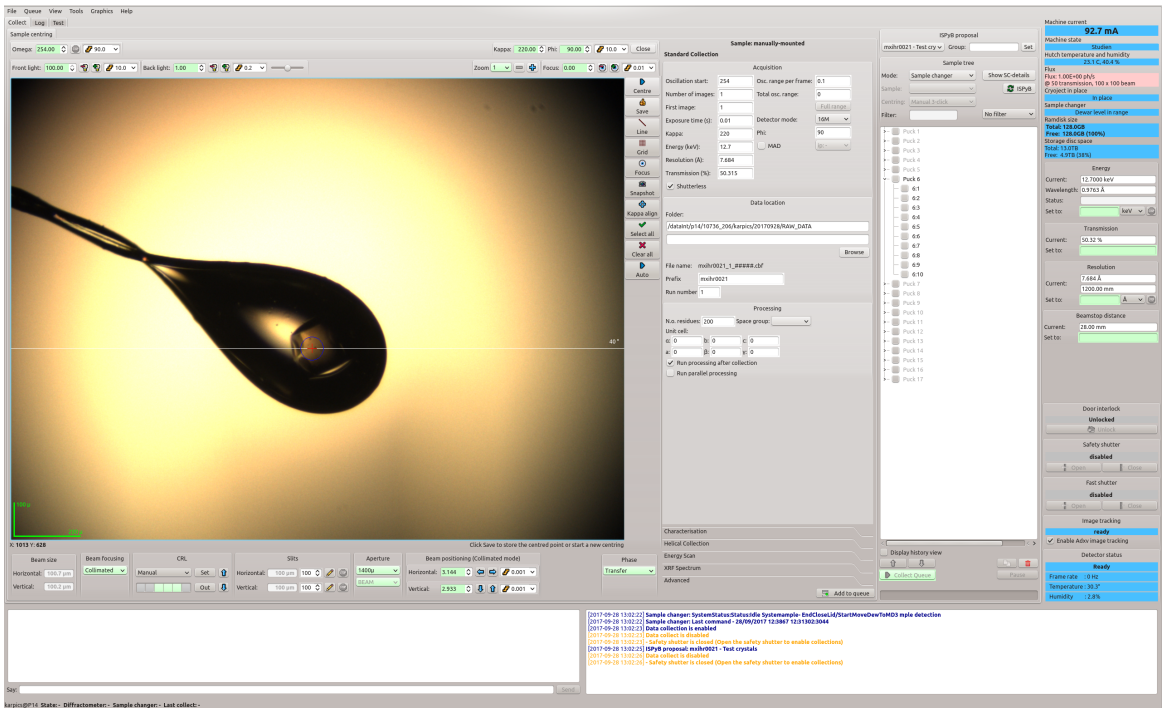

Figure 1: Graphical user interface MxCuBE as seen at EMBL beamline P14. In the left half, a life view of the sample is displayed. The sample view is surrounded (clockwise from above) by controls for sample orientation and microscope settings, controls for sample centering procedures, and widgets for controlling the size and shape of the X-ray beam (which need to be modified to match the size of the sample). In the right half, the left section offers several tabs that allow the definition of parameters of relevance to various data collection procedures, the middle section controls the loading and unloading of samples and the assembly of sequences of data collection procedures, while the right section allows to monitor (and partly control) general properties of the synchrotron, the experimental environment etc. At the bottom of the screen logging information is displayed.

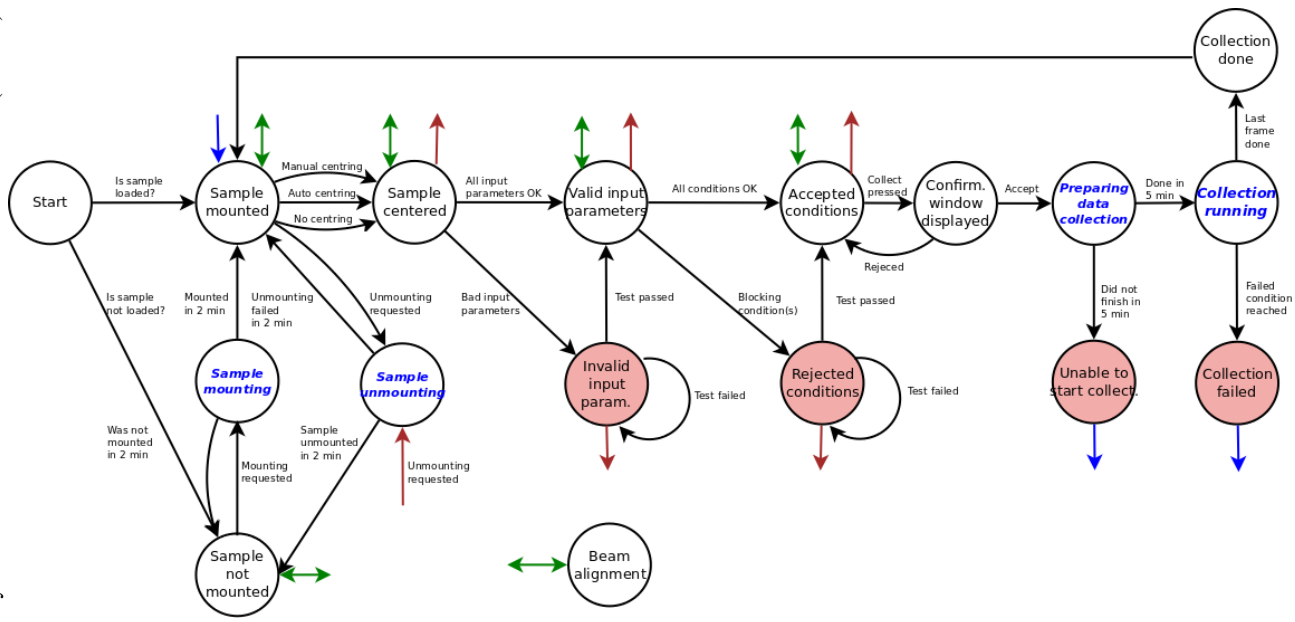

Figure 2: State graph of a user interaction during a macromolecular crystallography data collection. Circles represent FSM discrete states. Normal states are shown in white, while states painted in red are error states and require actions to return system to a normal state. Blue arrows point to *Sample mounted* state and are executed automatically, red arrows indicate the request from a user to unmount a sample. FSM graph includes also *Beam alignment* state where several xray beam alignment procedures are allowed.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA187

Based on these transitions, a state graph was created (Fig. 2). The graph originates from the *Start* node. If there is a sample mounted on the goniometer, then the state is changed to the Sample mounted state, otherwise to the *Sample not mounted* state. From the *Sample not mounted* state it is possible to reach the *Sample mounted* state via *Sample mounting* state (an attempt to mount a sample). In case the of successful mounting, the system reaches the *Sample mounted* state while for a failed mounting procedure it resorts to the *Sample not mounted* state. The next state, *Sample centered*, is reached from *Sample mounted* state via three different transitions aka positioning methods (manual, automatic optical, or xray based centering). To reach the state in which data are actually collected - *Collection running* - the system has to go through several preparatory stages. These include reaching a set of *Valid input parameters*, arriving in a set of *Accepted conditions*, and leaving the *Confirmation window displayed* state with a positive confirmation. To ensure a closed loop system most of the states have transitions to the *Sample mounted* state.

IMPLEMENTATION IN THE GRAPHICAL USER INTERFACE MXCUBE

Both P13 and P14 use MxCuBE with Qt4 graphics [6, 7] as an experimental graphical user interface. MxCuBE is written in Python and is logically divided into a hardware access and a graphical representation layer. The hardware access level is called 'Hardware Repository' and contains a set of configurable hardware objects. The modularity of the Hardware Repository allows sharing data across and between hardware objects and creating complex so called 'procedures' to perform scans, data collections, beam centering and other complex processes. We have implemented the FSM in MxCuBE as a hardware object that can connect to all hardware objects relevant to performing measurements or experiments. The implementation of the FSM-graph is defined via an yaml-file describing graph nodes, transitions and conditions. Transitions are triggered upon request when the conditions as evaluated by individual hardware objects are fulfilled. For example, the *sample_is_loaded* condition (Fig. 2) is broadcasted by the goniometer hardware object. For debugging purposes a window with information about the state of the FSM is available. It contains a table displaying all available states, conditions and a history of states assumed by the system (Fig. 3).

CONCLUSION AND PERSPECTIVES

We have presented an idealized Finite State Machine model for the interaction of a beamline user with a beamline to collect diffraction data from a crystal. While analyzing the current implementation of the process in the MxCuBE graphical user interface running on the MX beamlines at EMBL-Hamburg, we realized the high complexity of the real-world interaction between a user of a beamline. When restricting the FSM-model to the steps relating to a simple measurement on a single sample, the resulting description is helpful for the

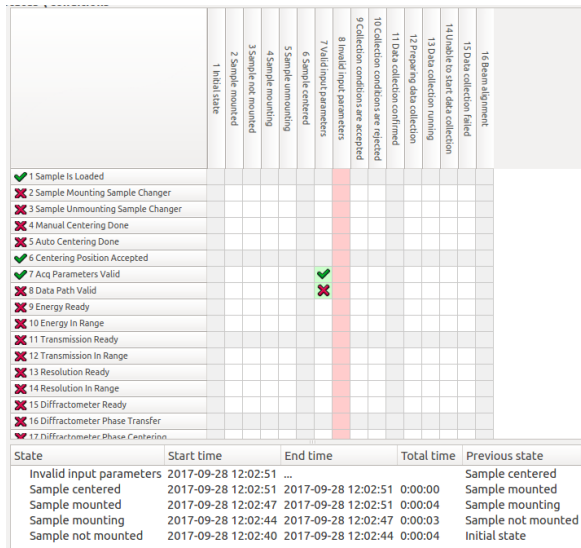

Figure 3: Window for monitoring the FSM described in (Fig. 2) as implemented in MxCuBE. In the matrix in the upper part of window, each column corresponds to a state and each row of the matrix relates to a condition that needs to be fulfilled for a transition between two states to be executed. For each condition, the corresponding state is indicated by a green tick-mark (true) or a red cross (false). At the bottom, a state history is displayed. In this example, the sample was mounted and successfully centered: the FSM has progressed through *Sample not mounted*, *Sample mounting*, *Sample mounted* into the *Sample centered* state. According to the FSM graph, the system can now progress to *Valid input parameters* or to *Invalid input parameters*. As the condition *Data path valid* is not fulfilled system is in the error state *Invalid input parameters* (column 8 painted in red). To progress towards the *Data collection state*, the system has to go through the *Valid input parameters* state which can be achieved by entering correct data collection path.

user of the beamline, the beamline scientist supporting the beamline user, and for the developers of the beamline control interface. For the (inexperienced) beamline user, being informed about the current state/current transition is useful especially in situations in which the beamline is seemingly idle or blocked, e.g. during a technical wait period when a sample is robotically mounted. The clean information about error state and - when possible - suggested recovery procedures make the beamline user more autonomous and reduces the need for the beamline scientist to recover the beamline. Error situations unrecoverable by the beamline user can also be recognized and an automatic notification can be sent to the beamline scientist to address the problem. For the developer, the state history provides an important tool for debugging. Gathering statistics about the behaviour of beamline components as seen via the states assumed and transitions take can be used to build a knowledge base for pin-pointing fault-causing beamline components. During the design and implementation of the FSM, a number of

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA187

conceptual challenges surfaced. In particular, the possibility of accessing many beamline functionalities from the same interface-section at the same time will need be reassessed. A hierarchical approach could be envisioned in which actions that can be triggered by the beamline are cleanly grouped so that only a subset of functionalities - that is less complex to maintain in a consistent way - is accessible, while the other functionalities are blocked. Blocked functionalities can be easily indicated by 'greying-out' the respective parts of the GUI. While the description of an entire beamline as an FSM may be conceptually of interest, the involved effort may be prohibitive. Nevertheless, describing subsystems as FSMs, such as done here for the interaction of the user with the beamline for a standard diffraction data measurement, can be useful both for achieving a better understanding of the needs and for optimizing procedures in terms of efficiency and robustness.

REFERENCES

[1] J. Gabadinho *et al.*, "MxCuBE: a synchrotron beamline control environment customized for macromolecular crystallography experiments", *Journal of synchrotron radiation*. vol. 17, pt. 5, pp. 700–707, 2010.

[2] D. Harel, "Statecharts: a visual formalism for complex systems", *Science of Computer Programming*, vol 8, iss 3, pp. 231–274, 1987.

[3] F. Calheiros, P. Golonka, and F. Varela, "Automating The Configuration Of The Control Systems Of The Lhc Experiments", in *Proc. of ICALEPCS2007*, Knoxville, USA, October 2007, paper RPPA04, pp. 529–531.

[4] G. De Cataldo, A. Augustinus, M. Boccioli, P. Chochula, and L. Stig Jirdén, "Finite State Machines for Integration and Control in ALICE", in *Proc. of ICALEPCS2007*, Knoxville, USA, October 2007, paper RPPB21, pp. 650–652.

[5] B. C. Heisen *et al.*, "Karabo: An Integrated Software Framework Combining Control, Data Management, And Scientific Computing Tasks", in *Proc. of ICALEPCS2013*, San Francisco, USA, October 2013, paper FRCOAAB02, pp. 1465–1468.

[6] I. Karpics, G. Bourenkov, M. Nikolova, and T. R. Schneider, "Graphical user interface and experiment control software at the MX beamlines at EMBL Hamburg" in *Proc. of NOBUGS (New Opportunities for Better User Group Software), NOBUGS2016*, Copenhage, Denmark, October 2016, paper 10.17199/NOBUGS2016.91, pp. 53–58.

[7] M. Cianci *et al.*, "P13, the EMBL macromolecular crystallography beamline at the low-emittance PETRA III ring for high- and low-energy phasing with variable beam focusing", *Journal of synchrotron radiation*, vol. 24, no. 1, pp. 323–332, 2017.