ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-WEBPL04

SOFTWARE ARCHITECTURE FOR BEAMLINE AUTOMATION – VMXi USE-CASE

C. J. Sharpe, Diamond Light Source Ltd, Oxfordshire, UK.

Abstract

Versatile Macromolecular in-situ (VMXi) is the first beamline at Diamond Light Source (DLS) to be entirely automated with no direct user interaction to set up and control experiments. This marks a radical departure from other beamlines at the facility and it has presented a significant design challenge to General Data Acquisition (GDA), the in-house software that manages beamline data collection. GDA has become a reactive controller for continual, uninterrupted processing of all user experiments. A major achievement has been to demonstrate that it is possible to successfully deliver a suitable architectural implementation for automation developed within a standard integrate development environment (IDE). There is no need for specialised software or a domain specific language for automation. The objective is to: review VMXi project with the emphasis on hardware configuration and experiment processing; describe the software and control architecture for automation; and provide a general set of guidelines for developing software for automation at a scientific facility.

VMXI OVERVIEW

In 2013 I02 - one of several MX [1] beamlines at DLS [2] - was selected for a major hardware upgrade. The scientific driver for the new beamline was to make it specialised for *in-situ* diffraction of crystallised macromolecular samples [3]. Multiple crystals of purified samples are grown under different chemical conditions within wells on crystallisation plates. Crystallisation experiments usually fall into two distinct stages: screening and data collection. Screening involves exposing a plate to X-rays to discover if any samples diffract. Data collection aims to maximise diffraction information output by applying a wide range of X-ray parameters on plate samples. One plate has many samples so that a great volume of data can be generated and processed in a concentrated period of beam time. A plate is designed for high throughput and robot handling. The new beamline VMXi [4] fully exploits this so that many plates can be stored at a time and continually processed, vastly increasing experiment throughput and optimising beam use. A schematic of the VMXi end station is shown in Fig.1 without the radiation shielding. Plates are sent to VMXi are stored in two temperature controlled storage units (Rock Imager 1000, Formulatrix) each storing up to 750 plates. Plates are conveyed through the radiation shielding to a local storage area within the data collection environment. This storage area can hold up to 12 plates and serves as a buffer for quick loading and unloading of plates to the goniometer.

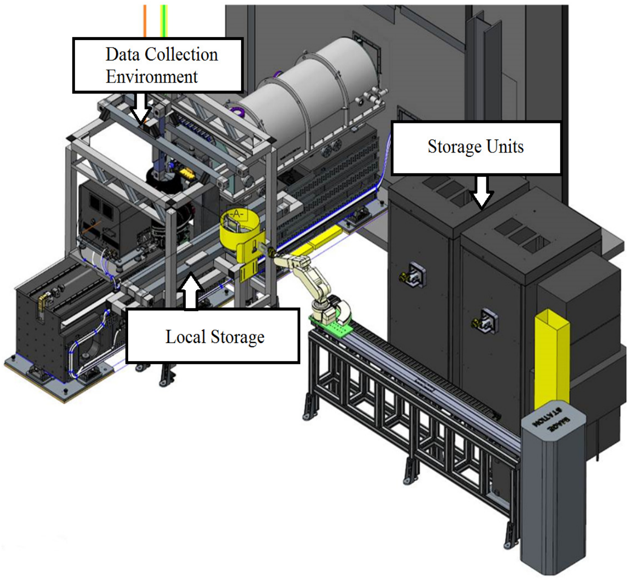

Figure 1: VMXi End Station.

The data collection environment hardware consists of a bespoke goniometer developed at DLS that holds a plate with submicron precision; the latest detector technology (Eiger 4M, Dectris AG); a fast shutter (operating at 4 milliseconds opening time); and a retractable, high resolution On Axis Viewing system (OAV) for sample imaging. VMXi is a high flux micro focus beamline to achieve high sample throughput. In-situ experiments are conducted at room temperature and when crystals are exposed to X-rays they are destroyed in milliseconds. The high specification detector and fast shutter optimise the diffraction information captured under these conditions.

DATA ACQUISITION SOFTWARE

The GDA [5] framework is software developed using Eclipse Java IDE [6] and deployed across beamlines at DLS to enable a high level interface for users to conduct experiments and collect data. A user runs experiments through GDA on site or (as is the case on MX beamlines) off-site through a remote client.

GDA is an implementation of a client server model. The client is an Eclipse RCP that provides a graphical interface through which a user can control and move beamline hardware and execute experiments. The server has many components but the main features to highlight are: Java Channel Access plugin to communicate with EPICS IOCs [7] (the hardware control layer); Java objects that are an interface to beamline hardware required for experiments; and a Jython server to run and execute scripts written in Python syntax. Taken as a whole scripts define the operational functionality of a beamline and as such are a crucial resource; in essence they are high level management and control of the beamline for experiments. Scripts are written by software developers, beamline staff, and users.

AUTOMATION

A fundamental requirement for VMXi was automation. In terms of an operational objective this means continual processing of plates without interruption for several weeks at a time - as long as there are plates available to be processed. Although not explicitly stated, it was assumed that GDA could be readily adapted to automate a beamline. However, VMXi breaks the client server model as there is no direct, dynamic user interaction with the beamline. Synchweb [8] is a web interface used at DLS through which users define data collection experiments. This information is persisted in ISPyB [9], a database developed specifically for the MX experiment pipeline. Users have no further involvement in the experiment after a plate is scheduled for data collection. At some point in the future, GDA would manage the conveyance of a scheduled plate to the beamline and process it according to the user experiment parameters in ISPyB. GDA must perform this task repeatedly, indefinitely, and without failing.

ARCHITECTURAL MODEL

Software design for automation falls squarely is in the domain of Artificial Intelligence (AI) [10,11]. It is a mature technology that has been successfully applied to many different and challenging real world control problems - this includes automating industrial manufacturing processes. There is nothing new or especially novel in the management and of control VMXi - or any similar beamline for that matter - that prevents it functioning as an industrial process by application of the same AI principles. All AI architectures are an arrangement of three basic capabilities, Sense, Plan and Act (SPA), which enable an agent to function autonomously in its operational environment.

- Sense – excitation of a sensor attuned to sense the state of some artefact in the environment and feeding sensed state data in to an agent.
- Plan – Updating the state of a global plan with the sensed data, and use this plan to generate a control action.
- Act - present the control action to the agent actuators and do the action.

SOFTWARE DESIGN

The overall high level control and management of VMXi is framed conceptually in terms of SPA. The architectural design is a specification of these basic building blocks and how they are wired together.

High Level Planning

A Data Collection Plan is encapsulated in user experiment parameters, including the state in to which the beamline needs to be set to perform an experiment. This plan is persisted in ISPyB and is required only when the plate associated to it is loaded and ready to be shot. What plate to load from the storage unit when there are hundreds plates in a scheduled state is a dynamic and complex process that forms a Load Plan. There is a wide criterion that informs how this Load Plan is ordered and reordered, but the reasoning is not important to GDA, only the end result - the plate to load next. These plan elements are the Plan component, which is external to GDA. What is required is to get Plan information into GDA to drive its actions.

Low Level Control

The complexity of an environment indicates the extent of low level control and real time reactiveness required by GDA to deal with sudden and unexpected events; this does not require sophisticated behaviour. (As it can be observed from the natural world, simple creatures function perfectly well in a highly complex, dynamic environment without higher level cognitive ability). In VMXi any hardware that can be directly controlled by GDA is considered a part of the internal environment. Any resource required by GDA to function is a part of the external environment. In general GDA must be robust and act in a timely and controlled manner to any fault that may occur in either environment. GDA initialises a control action and monitors the progress of the action, but it is not responsible for performing action execution and low level control.

Agent Design

Moving the Plan component out of GDA leaves the Sense and Act components to be wired together. This creates a Reactive Controller, a standard AI architectural model. The model and model integrity provides a blue print for agent design. Its attributes are:

- A tight coupling between what is sensed and the action generated.
- It is stateless - there in no internal representation of the world.
- It should not execute any computationally difficult or intensive task that is an impediment to agent reactiveness.

The overall robustness of GDA is greatly enhanced by this model. If there is a catastrophic failure in the system (for example the server on which GDA is hosted goes down), agent performance is unaffected after recovery and restart. There is no state to recover and reconstruct, an agent will behave exactly as it did before the failure.

VMXi is logically separated into two distinct tasks: one task to manage the movement of plates; and another task to set up the beamline and collect data from a plate. Accordingly there is an agent dedicated to perform each respective task. Agents are situated in the internal environment as previously described; they initiate and control what and when actions are initiated and by extension a pre-determined response can be implemented to deal with any failure to complete an action.

Sample Handling Agent

Sample handling involves synchronised co-ordination of several separate pieces of hardware to move plates around the beamline. The micro-management and co-ordination of sample handling hardware is the responsibility of the low level control layer. An interface to the low level control program is a set of four EPICS process variables (PVs) that represent high level sample handling actions.

- Load from a storage unit to local storage.
- Load from local storage to goniometer.
- Unload from goniometer to local storage.
- Unload from local storage to a storage unit.

All high level sample handling actions implement the same control sequence:

- Set the PV to trigger the low level control program with plate source and/or plate destination parameters if required.
- Monitor the running state of the low level program.
- Set a flag to indicate the low level program has finished.

The low level program does not return a value after it finishes. Additional steps are added to check that any of the sample handling hardware is in a fault state.

- Set PV to retrieve the sample handling fault state.
- If a fault value is returned set a flag to indicate a hardware fault has occurred.

The Sample Handling agent determines which of the four actions to select at any given point in time and ensures exactly one action at a time is executed. Brooks's subsumption architecture [12] is the controller for the Sample Handling agent. It defines the agent as a hierarchy of behavioural layers. The higher the behaviour layer the greater the priority of its behaviour – in the terminology higher level layers subsume lower level layers. Behaviours are entirely modular, which makes it straight forward to incorporate new behavioural layers into the controller. This architecture is recommended where the overall behaviour of the controller is dedicated to doing a single task composed of a small number of behavioural layers.

Sensors data is input to behaviour layers in parallel. A behaviour layer is then said to be 'fired' if one of its sensor inputs has been excited. The sensed environment state is matched against a set of conditional rules that map to one candidate action from a finite set of actions for the behaviour. The rules are hierarchical and entirely deterministic. The order of these rules does have an impact on the emergent behaviour of the controller. If two behaviour layers are fired at the same time and two actions are generated, the higher level behaviour supresses the input of lower level behaviours and its actions is presented to the actuator. An inhibit signal can be fed into to a sensor so its output is prevented from firing a behaviour layer.

The Sample Handling agent controller shown in Fig. 2 is an adaption of the subsumption architecture.

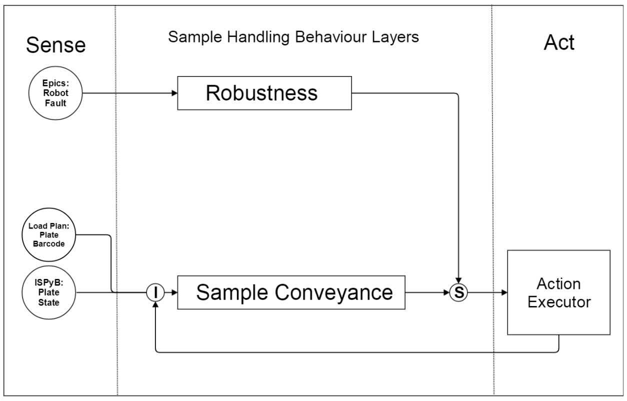

Figure 2: Sample Handling Agent Controller.

Sample Conveyance is the lowest level behaviour layer. Its goal is to convey plates to and from the beamline. Sensor inputs actively get state data from two information sources. The Load Plan service returns a plate barcode of the plate to load next into local storage. This information is used to retrieve from ISPyB plate state details for that plate. ISPyB is also requested to return plate state details of all plates in the data collection environment. Sensor input is fed concurrently to Sample Conveyance. A Sample Conveyance rules layer maps the sensed state to a rule in its rule set that represents one of the four high level sample handling actions.

The Action Executor executes sample handling actions passed in to it and monitors whether the low level program finished flag has been set. The action is executed on a separate processing thread which allows for the main controller to gather Sensor data while an action is executing. Before an action is executed all Sensor output is inhibited to all behaviour layers. This guarantees that one conveyance action is executed at a time. After the action has executed all Sensor output is uninhibited.

Plate location state is updated in ISPyB just before the Action Executor executes a conveyance action, and immediately after a conveyance action has successfully completed. It is critical plate location state matches its physical location. If plate location state is not correct then Sample Handling agent may select an undesirable action.

Robustness is the highest level behaviour layer. It protects plates and the sample handling hardware in a controlled and robust manner in the event of an internal fault caused during sample conveyance. An error value is set at the controls layer if there is any fault event at any point during sample handling. The Robustness rules layer executes a set of actions to generate a fault report, gathering relevant state information from the low level sample handling controllers. This fault report is then broadcast to support staff. Input layers are inhibited and the controller is safely stopped. When the fault is rectified it is then a manual process to physically move a plate to the correct location, and ensure the plate's location state is consistent with that in ISPyB. The controller is manually restarted, and the input signal to the Conveyance layer uninhibited by support staff.

Data Collection Agent

Data Collection first puts the beamline in to a state ready for data collection and then simultaneously moves the detector, shutter and goniometer to collect data. Hardware is tightly packed in the data collection environment and has a potential for a collision. To avoid collisions a set of well defined-routines move hardware in the mini-hutch to pre-set safe positions for a given task, each routine verified through exhaustive testing.

It is difficult to write concise code to synchronise the movement of hardware so that it collects data at the correct place and at the correct time. As previously stated, GDA should initiate and monitor actions and no more. VMXi utilises a part of a major project [13] developed at DLS that pushes configuration, management and synchronisation of hardware for data collection in to hardware controllers.

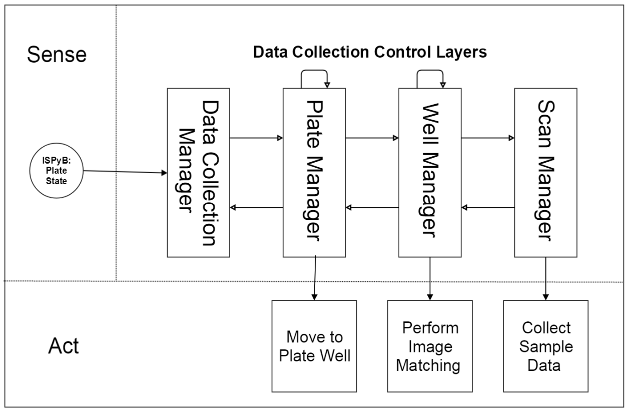

Figure 3: Sample Handling Agent Controller.

The data collection agent controller shown at Fig. 3 is structured around a data collection algorithm specified by beamline staff. A layer is a subset of steps in the algorithm; they are not hierarchical and are processed in order. The beamline hardware is moved to a set safe state at each layer as required by the action to be performed. After one layer has completed its steps in the algorithm, control is passed on to the next layer.

A Sensor gets plate state from ISPyB. The sensed state is an input in to the Data Collection manager layer. If a plate is sensed to be on the goniometer and it is ready to be shot, this initiates a data collection. All the user experiment parameters for the plate are requested from ISPyB. When they are returned they are passed in to the Plate Manager layer. The user experiment parameters are grouped by well. The Plate Manager is then responsible for moving the plate to each well on the plate. After moving to a well, the well data collection parameters are passed in to the Well Manager. Its role is to generate the real motor and hardware controller values for each data collection in the well. A plate may have been stored for weeks at a time before data collection. In that time a crystal may have grown. A crystal may move during sample handling, or through the effect of gravity as plates are kept upright in local storage. Image matching is performed on the original well image and the well image taken from the On Axis Viewing Camera. It checks the quality of a match between the two images and generates a delta. This value is applied to the original data collection points marked by a user on the image to correct for crystal displacement. Computationally this is a very intensive task. To keep model integrity for a reactive agent, Image Matching is made an external service. Individual data collections are then passed in to the Scan Manager. The hardware controllers are set up for a data collection. The Scan Manager sends a signal to the hardware controllers to collect data. After this is completed, control returns to the Well Manager and the next data collection in the well is processed.

After all data collections in a well are processed, control returns to the Plate Manager. It moves to the next well on the plate - and so on and so forth until all data is collected on the plate and control returns to the Data Collection manager. The plate is then ready to be unloaded from the goniometer.

Data collection agent robustness is complicated by the fact that it is dependent on a variety of software and hardware resources internal and external to GDA. There is no one uniform response to deal with faults and failures. The decision is on a case by case basis. A general rule is that before and after performing some task a check is made of the state of any resources used by the task. If a resource is in a fault state or is unavailable then that task should not continue and will safety terminate. Control passes back to the Data Collection Manager layer and the Data Collection agent stops. The data collection is manually restarted by support staff when the fault is corrected or the resource becomes available.

Agent Task Coordination and Communication

The Data Collection agent and Sample Handling agent act independently of one another, but both are reliant on sensing plate location and X-ray exposure state to determine what action to take. There is no centralised control of either agent but rather a centralised information resource for inter-agent communication. Plate state is persisted in ISPyB as an attribute of a plate. Updating plate state is an indirect signal, a perceptual cue as it were, for a change in agent behaviour. This can be represented as a state machine shown in Fig. 4. The Sample Handling agent is responsible for updating location state of a plate, and the Data Collection agent the X-ray exposure state. The operational advantage is there is the no detrimental impact on performance if an agent is not running or restarted – it does not need to have a memory of past actions to know what action do next, only select an action based on sensing the current plate state.

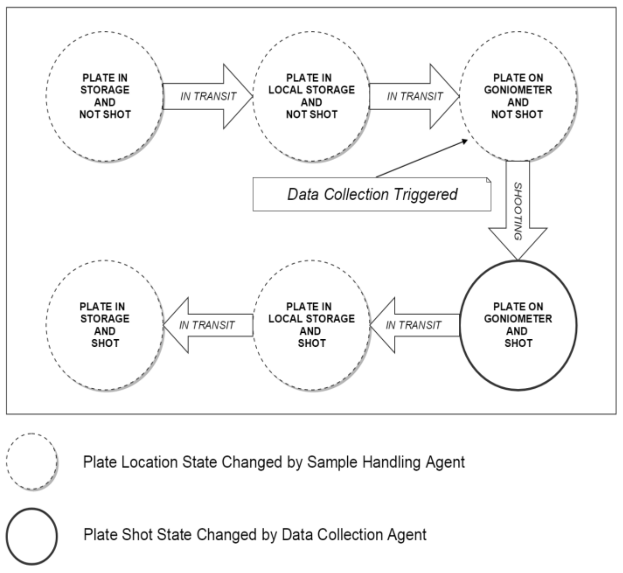
PLATE IN STORAGE AND NOT SHOT
IN TRANSIT
PLATE IN LOCAL STORAGE AND NOT SHOT
IN TRANSIT
PLATE ON GONIOMETER AND NOT SHOT
Data Collection Triggered
SHOOTING
PLATE IN STORAGE AND SHOT
IN TRANSIT
PLATE IN LOCAL STORAGE AND SHOT
IN TRANSIT
PLATE ON GONIOMETER AND SHOT
Plate Location State Changed by Sample Handling Agent
Plate Shot State Changed by Data Collection Agent

Figure 4: State Machine for Plate State.

SOFTWARE IMPLEMENTATION

The core components of GDA implementation for VMXi are shown in Fig. 5.

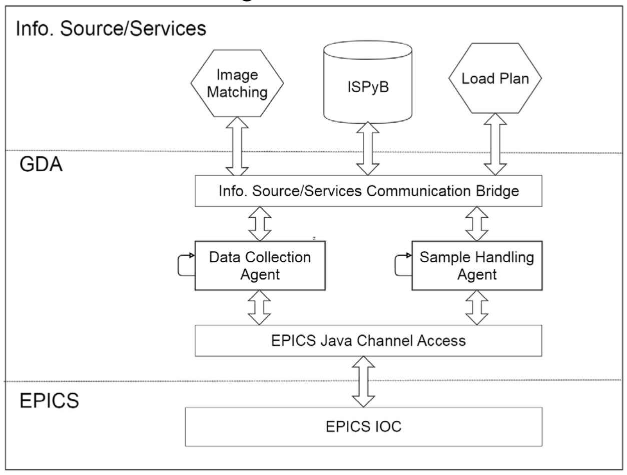

Figure 5: GDA for VMXi.

The main control structure of a reactive controller is a loop that cycles at a set time period, processing Sense and Act in that order. The Sample Handling agent and Data Collection agents are built around this very simple structure. Agent architecture is implemented as scripts written in Python syntax. The code is highly modular and its foot print is very small (under 3000 lines). This clearly shows that a seemingly complex task does not require proportionately complicated code, just the correct design model. Each agent is run on separate processing threads within the GDA server. Channel Access to the EPICS control layer remains unchanged.

Communication Layer

Several Java plugins have been developed to serve as a communication bridge between GDA and any external information resources or services it needs. A try-retry policy is put in place at each communication point to guard against connection failure. For critical write sections to update plate state in ISPyB the agent does not time out and waits indefinitely until a connection is re-established. The additional network traffic generated is minimal, and does not impact on the performance of the database server that is shared by other MX beamlines.

External Services

The Load Plan and Image Matching services are started as separate processes and communication to the external services is handled via an ActiveMQ server. The Active MQ server and Load Plan run on the same machine as the GDA server. The Image Matching service runs on another machine.

GDA Client and Manual Interaction

A minimal GDA client provides a manual interface for beamline support staff. Through the command terminal view in the client agent processes are started and stopped on the server. It is also a tool for running the extensive set of unit and integration tests. Plate state (and hence the state machine) can be set in ISPyB from the client. Future development is to add features to show the internal state of agents and the state machine.

Logging and Fault Reporting

Concise logging is essential to trace the actions of the agent and beamline state at any given moment in time, especially in the event of a fault being detected that result in the agent stopping what it is doing. VMXi will operate day and night, and support staff may not be immediately available to deal with a problem. It is essential to log what has caused the fault and structure this information as a formal fault report. A fault report may also give instruction on how to correct the fault and return both hardware and software in to a good state. For example, the robot load fails in sample handling; the error is detected by the Sample Handling agent. As a part of the Robustness Layer behaviour a fault report is issued. Within that report are detailed the nature of the error and the manual steps to reset the robot and the plate state in ISPyB. The Sample Handling agent then stops after this report is issued.

Performance Testing

VMXi has entered an extensive commissioning phase after reaching the major project milestone of a first external user. In that time the new architecture for GDA has shown that it meets the demands for beamline automation and satisfies the global objective of VMXi. Plates have been processed for hours at a time, uninterrupted in several separate commissioning tests. Agents were started, stopped, and re-started and carried on their task without causing disruption to the system as a whole.

A significant amount of new hardware is being installed on to VMXi. With this comes new functionality to integrate into the software. Once hardware and software stability is reached, the focus will then shift to optimising beamline performance and throughput. The inherent modularity of the software allows it to be re-structured so it can be subjected to formal performance tests. This need not be a 'live' test, which may not be possible if maintenance work is undertaken at the beamline. A fully fledged simulator used during the development of the software can be utilised for this purpose to run formal tests offline. The simulator output would produce statistical evidence of the impact a great deal of work to improve this. Here too the simulator can be employed to test agent robustness to by introducing random faults to the agent at any point while it is running, prove that does not fall over, and each aspect of it robustness behaviour is exactly as expected.

using the development platform used by GDA developers. There is no

beamline staff: Thomas Sorenson, Juan Sanchez-Weatherby, James Sandy and Carina Lobley. Scientific Software: Alun Ashton, Kevin Savage, Karl Levik, Stuart Fisher and Graeme Winter. Controls: James O'Hea, Lee Hudson, Alan Greer, Kris Ward and Chris Martin. GDA: Paul Hathaway. And to all the mechanical, electronical engineers and technicians who built VMXi.

REFERENCES

[1] GDA, http://www.opengda.org
[2] Diamond Light Source, http://www.diamond.ac.uk
[3] MX, http://www.diamond.ac.uk/Beamlines/Mx.html
[4] P. Aller, *et al.*, "Application of In Situ Diffraction in High-Throughput Structure Determination Platforms", In: Owens R. (Eds) Structural Proteomics. Methods in Molecular Biology, Vol 1261. Humana Press, New York, NY
[5] VMXi, http://www.diamond.ac.uk/Beamlines/Mx/VMXi.html
[6] Eclipse, https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/lunar
[7] EPICS, http://www.aps.anl.gov/epics
[8] S. Fischer, *et al.*, "SynchWeb: A Modern Interface for ISPyB", Journal of Applied Crystallography, Volume 48, Pages 927 – 932, June 2015.
[9] S.Delagenière, *et al.*, "ISPyB: An Information Management System for Synchrotron Macromolecular Crys-tallography".Bioinformatics, Volume 27, Issue 22, Pages 3186–3192, November 2011.
[10] R.R. Murphy, Introduction to AI Robotics, ISBN: 978-0262133838.
[11] S. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach Third Edition", ISBN: 978-1292153964.
[12] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, 2, Pages 14–23, 1996.
[13] M. Basham, J. Filik, "Generic Mapping Scans at Diamond Light Source", NoBugs, Copenhagen, Denmark, 17-19 October, 2016.