COMMITTEE ON DATA
CODATA
INTERNATIONAL SCIENCE COUNCIL

KadiStudio Use-Case Workflow: Automation of Data Processing for *in Situ* Micropillar Compression Tests

PRACTICE PAPER

ubiquity press

RIHAB AL-SALMAN
CAMILA AGUIAR TEIXEIRA
PHILIPP ZSCHUMME
SUBIN LEE
LARS GRIEM
JASMIN AGHASSI-HAGMANN
CHRISTOPH KIRCHLECHNER
MICHAEL SELZER

*Author affiliations can be found in the back matter of this article

ABSTRACT

Scientific processes produce huge amounts of data that are usually acquired, transformed and analyzed on a regular basis. Translating these processes into automatable and reproducible workflows is considered to be an efficient way to support scientists in performing repeated processes that would otherwise be time-consuming and error-prone tasks. Consequently, the quality of scientific research can be accelerated and enhanced. In this article, we present for the first time a use-case of KadiStudio as a tool to automate analysis procedures of scientific data that are repeatedly acquired from *in situ* scanning electron microscope (SEM) micromechanical testing. KadiStudio provides a desktop-based workflow editor as part of the ecosystem of Kadi4Mat: Karlsruhe Data Infrastructure for material Science. The presented workflow includes nodes for processing and analysis of different types of data, namely mechanical response in text format and a series of SEM images in video file format acquired during *in situ* SEM deformation tests. In addition, the raw and analyzed data are automatically uploaded to the KadiWeb repository via nodes based on the kadi-apy library.

CORRESPONDING AUTHOR:

Rihab Al-Salman

Institute for Applied Materials (IAM), Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany

rihab.al-salman2@kit.edu

KEYWORDS:

scientific workflow; Kadi4Mat; FAIR data principles; KadiStudio; automation; micromechanical testing

TO CITE THIS ARTICLE:

Al-Salman, R, Teixeira, CA, Zschumme, P, Lee, S, Griem, L, Aghassi-Hagmann, J, Kirchlechner, C and Selzer, M. 2023. KadiStudio Use-Case Workflow: Automation of Data Processing for *in Situ* Micropillar Compression Tests. *Data Science Journal*, 22: 21, pp. 1–11. DOI: https://doi.org/10.5334/dsj-2023-021

1 INTRODUCTION

Scientific progress, especially in the field of materials science, relies ever more on data processing and analysis, as well as high-performance computing environments. This involves computational tools and applications with complex data analysis and visualization steps. Therefore, scientists spend a lot of time understanding the logic behind the software; performing cycles of different tasks such as acquiring, transforming and analyzing the data; and storing/publishing the raw and analyzed data in a repository. Scientific workflows aim at automating these cycles in a way to make it easier for scientists to focus on their research rather than repetitive and complex computational tasks.

A scientific workflow is a well-defined sequence of sequential or parallel tasks related to a scientific process and their dependencies, and these tasks are processed as automatically as possible. For more information, we refer the reader to Deelman et al. (2009), Ludäscher et al. (2009) and Liu et al. (2015). There are several useful software systems that enable the creation, exchange and execution of workflows, such as Fireworks (Jain et al. 2015), AiiDA (Pizzi et al. 2016), Jupyter Notebooks (Kluyver et al. 2016) and Galaxy (Afgan et al. 2018). Griem et al. (2022) has recently reported on our new and openly accessible plugin framework called KadiStudio and its workflow editor, which is integrated within the Electronic Lab Notebook (ELN) functionality of Kadi4Mat (Karlsruhe Data Infrastructure for Materials Science) for creating, editing and executing scientific workflows. As mentioned in Griem et al. (2022), KadiStudio is developed to avoid some limitations associated with, for example, the earlier mentioned workflow systems, such as the need for good programming experience (AiiDA and Jupyter Notebooks) and the application in a specific domain (Fireworks and Galaxy). Therefore, KadiStudio offers a generic and domain-independent approach to formulate workflows and the data therein in a FAIR manner (the data must be findable, accessible, interoperable, and reusable) (Draxl and Scheffler 2020). In addition, Kadistudio targets not only scientists with an affinity for programming but also researchers with little programming expertise. The implementation of the workflow editor is based on an open source node editor library for the Qt GUI framework (Pinaev et al. 2017), offers a graphical frontend and hence is easy to use for scientists.

Herein, we present a scientific use-case of the workflow-editor plugin of KadiStudio. The workflow aims to automate data processing for *in situ* scanning electron microscope (SEM) micropillar compression tests (see more details in Section 2). Different *in situ* SEM deformation data are routinely acquired like: (1) the temporal evolution of the surface morphology by SEM videos/images, (2) displacements and (3) forces measured from a nanoindenter. Each micropillar compression test will have such data. Due to the underlying statistical nature of plasticity of the micro scale, it is highly recommended to test at least tens of micropillars for statistical interpretation. The acquired video, for example, must be analyzed by the scientist to correlate the changes in surface morphology of micropillars with its mechanical response measured by a load cell. Identifying changes in the slip morphology by manual processing of the videos consumes a lot of time, therefore automation of this analysis step makes the process more efficient and reproducible, as will be presented here.

2 BRIEF DESCRIPTION OF THE SCIENTIFIC PROCESS: *IN SITU* SEM MICROPILLAR COMPRESSION TESTS

In situ SEM deformation testing is a unique materials characterization technique designed to quantify the mechanical properties of materials at the micro- and nanoscale while observing the deformation behavior by electron microscopes. In the case of micropillar compression, it is possible to directly observe the deformation behavior of materials under uniaxial compressive stress and to measure the mechanical response. It has been widely used not only to characterize mechanical properties of nanomaterials but also to reveal fundamental deformation behaviors at small scales (Uchic et al. 2004). Typically, cylindrical pillars are fabricated by using a focused ion beam (FIB) microscope, and they subsequently deformed by a nanoindenter equipped with a flat-punch diamond tip inside a SEM (Uchic et al. 2004; Malyar et al. 2019). It enables the study of materials in small scale namely, in the order of a few micrometers down to hundred nanometers. This in turn allows the investigation of the size effect and/or the deformation mechanism in a particular region of the sample—for instance, a grain boundary or a specific grain orientation (Dehm et al. 2018; Malyar et al. 2019). In this study, we used equiatomic

FeCrCoMnNi high entropy alloy as a model material system, and we fabricated micropillars with a diameter of 2 micrometers. Finally, the pillars were tested inside a SEM (Merlin, Zeiss GmbH) using a nanoindenter (Hysitron PI89, Bruker).

A remarkable advantage of this testing technique is that during the *in situ* SEM testing, videos of the microcompression test, as well as load versus displacement curves, are simultaneously recorded. Thus, the deformation behavior under the imposed stress condition can be directly observed while acquiring mechanical properties such as yield stress and critical resolved shear stress (CRSS) (Dehm et al. 2018). The possibility to observe the material deformation as it occurs allows to pinpoint (1) which slip system is activated and (2) applied stress at the moment when the slip activated. To do the correlative analysis, one should analyze each frame from the video recorded during testing and observe the moment the events (deformation of the micropillar/slip morphology) occur. A typical micropillar compression test takes around 60 seconds, of which 15 frames are taken per second, adding up to a total of 900 frames that take roughly 30 minutes to be analyzed. Due to large scatter on the mechanical data once samples are reduced in size (stochastic behavior) (Dehm et al. 2018), a large number of micropillars per condition should be tested to obtain statistically relevant mechanical data. Therefore, applying an automatic recognition to precisely identify the events and correlate them to the data is extremely helpful for an otherwise tedious and time-consuming task.

3 KADISTUDIO USE-CASE WORKFLOW

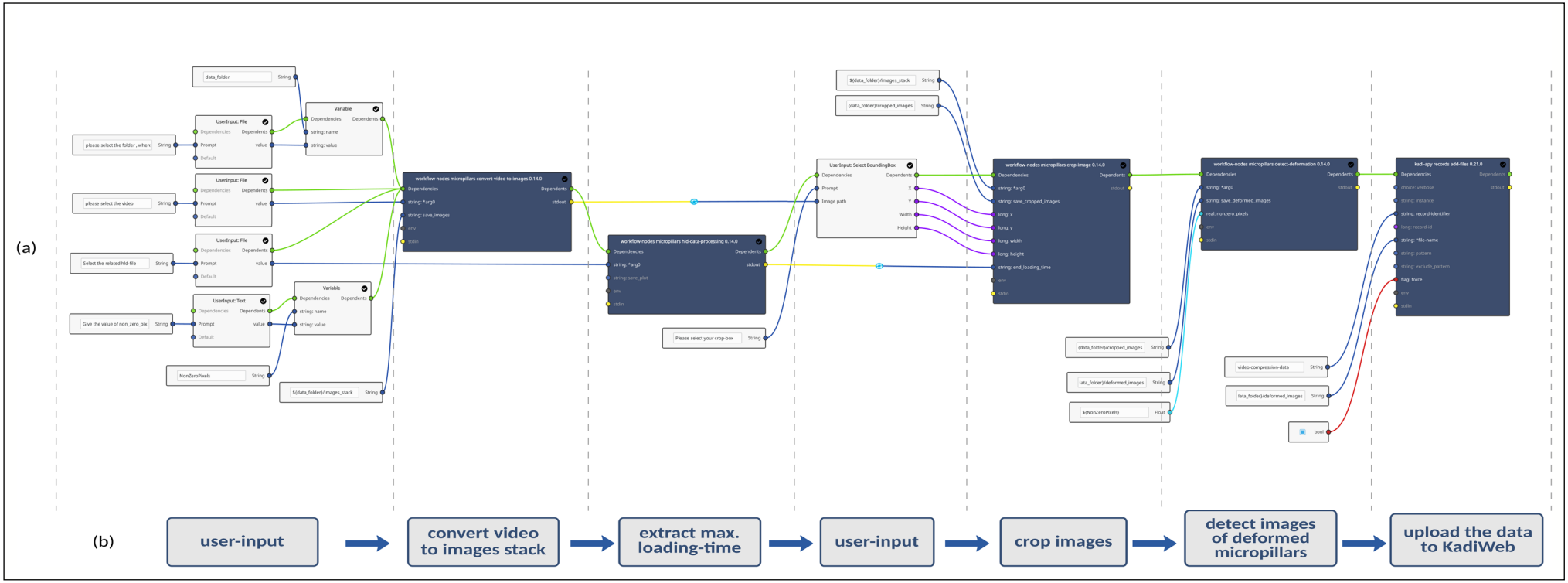
3.1 BACKGROUND

As described in detail in Griem et al. (2022), the creation of workflows in KadiStudio is based on dividing the scientific process into individual functions or tools that are considered to be the basic building blocks of the process. Each building block represents an atomistic, that is, impartible subprocess that is implemented in KadiStudio in the form of an individual node based on the IPO (Input-Process-Output) model. The various nodes are added and connected within a graphical user interface (GUI) using an intuitive point-and-click mechanism. Each of the insertable nodes can be differentiated between (1) tool nodes, which serve to integrate various programs or functions that are customized according to the needs of the users; (2) environment nodes, which are used in combination with tool nodes and allow the execution of a tool node in another environment like a remote-computer or a device; and (3) built-in nodes, which allow one to influence the execution of the workflow and add interactive options as well as variables, loops, if-condition and so forth to the editor.

According to the IPO model, each node has input ports, which are located on the left side of the node, and output ports on the right side (see Figure 1). Depending on their task, the ports can be divided into parameterization, dependency, environment and stdin/stdout ports. Stdin and stdout ports refer to the standard input and standard output streams of the underlying program, respectively. Parameterization ports are used to pass arguments and options necessary to execute the node, such as string or Boolean values. The execution order of the nodes can be defined using the dependency ports.

When executing a workflow, the added nodes and their connections are processed and, in the case of tool nodes and environment nodes, translated into command line interface (CLI) commands. As mentioned in Griem et al. (2022), to implement a new node for KadiStudio, the underlying CLI command must be (1) executable and (2) provide the ––xmlhelp option. This means that any script (regardless of the programming language) that fulfils these two conditions can be implemented in KadiStudio. The ––xmlhelp option returns a machine-readable description of the command, which is needed by the workflow editor to create the visual representation of the command within the editor. In case the needed tool does not provide this option, it can be added retrospectively with a wrapper script using the xmlhelpy Python library (Kadi4Mat Team and Contributors 2022c). Herein, all of the created tool nodes are based on python library xmlhelpy and can be added to the editor's tool list using its GUI. In addition, the underlying Python scripts of the tool nodes can be added to the workflow-nodes library (Kadi4Mat Team and Contributors 2022b), which already contains various Python-based nodes covering basic as well as more specialized functions.

Figure 1 (a) Visualization of the workflow created in the KadiStudio workflow editor, in order to automate data (SEM videos + compression data file) processing for microcompression tests. **(b)** Simplified flowchart representing the task of each segment of the workflow shown in (a).



3.2 AUTOMATION OF THE DATA-PROCESSING PROCEDURE FOR MICROPILLAR COMPRESSION TESTS

To automate the processing of the acquired *in situ* SEM videos and the other associated data of the microcompression test described earlier, a workflow is created for this purpose in the workflow editor KadiStudio. In the following, the workflow is divided into seven parts as illustrated in Figure 1, and for a clearer visualization, each part is displayed separately in Figures 2–8. This allows for a more detailed description of the single working steps and helps readers understand the dependencies between them by highlighting their inputs and outputs.

3.2.1 UserInput nodes

As described earlier, the tool nodes can be parameterized by connecting the input ports at the left side of each node with the desired input. This can be directly achieved by connecting these ports with the source nodes (String, Boolean, Integer, and so forth) inside the workflow itself. However, to make it even easier for the user, *UserInput* nodes are used for inputs that need different values for each individual execution of the workflow, so that the user is prompted to add his input in a clear manner through a dialog box that appears upon executing these nodes. In our use-case workflow, *UserInput* nodes are placed at the very beginning so the user provides all of the required inputs for all of the tool nodes at once. The inputs provided here are the path to the data folder, the MP4 video from the SEM machine, the microcompression data (text file containing data such as displacement, force, time, etc.) from the nanoindenter and the threshold value for the image pixel change (more details in subsection 3.2.6). This takes place directly after starting the workflow execution, as can be seen in Figure 2. Because some of the input values will be repeatedly used in the workflow, it is reasonable to store them in variables using the so-called *Variable* node (Figure 2a). The stored values can then be reused throughout the whole workflow by referencing the variable.

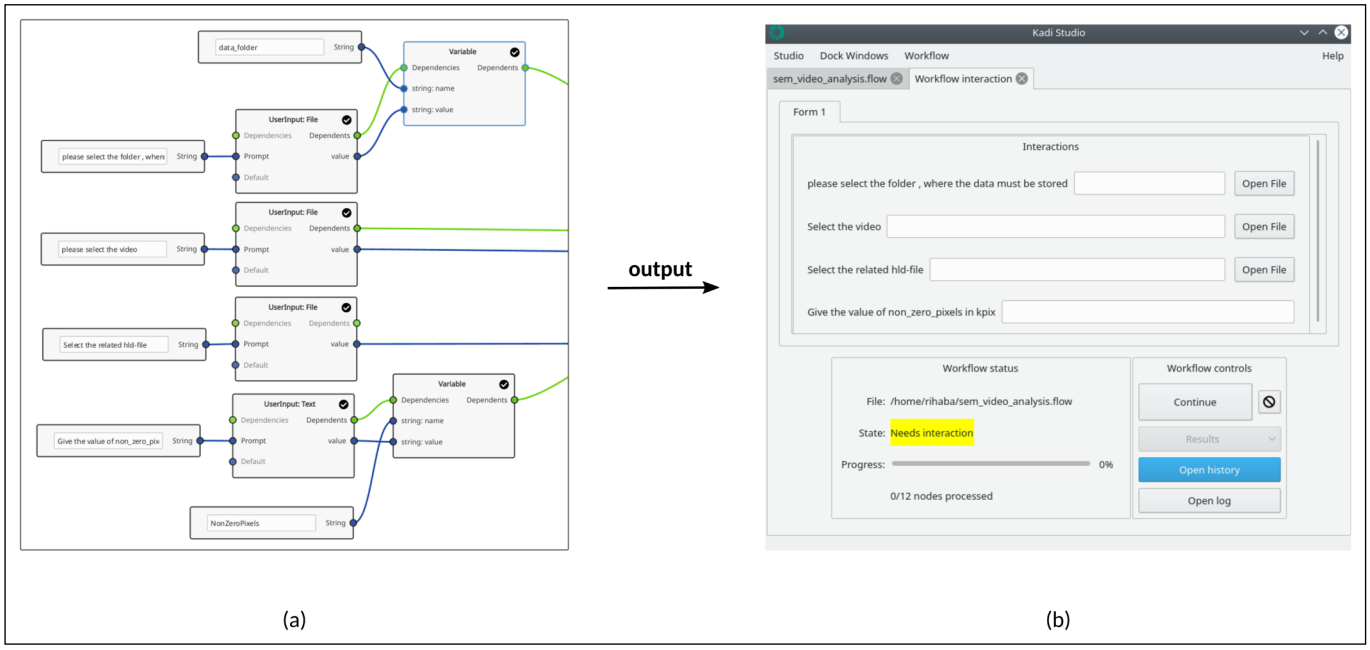
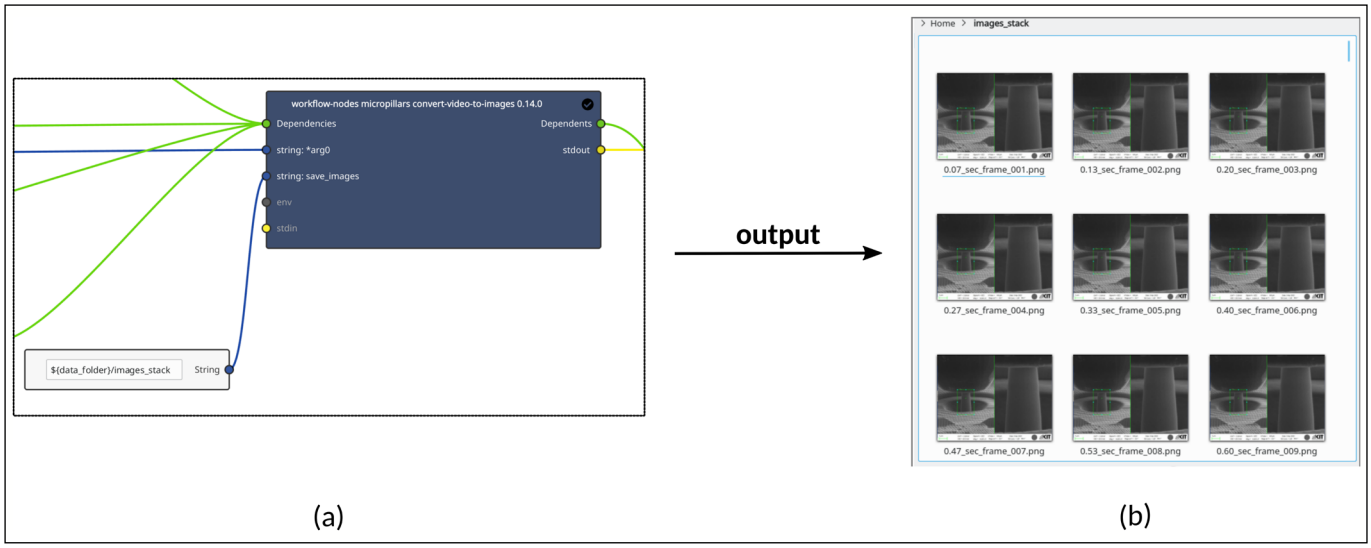


Figure 2 (a) User input nodes: *UserInput: File* and *UserInput: Text*, as well as *Variable* nodes. After execution of the user input nodes, the user is prompted through a dialog box **(b)** for an input, in this case to select the folder in which the outputs will be saved, the hld-file (a text-format file containing the micro-compression data), test parameters and the value of *nonzero pixels*. The folder path, as an example, is passed into a variable to be used several times in the workflow.

3.2.2 The convert-video-to-images node.

After adding the input values in the dialog box (Figure 2b), once the Continue button is clicked on, the next tool nodes will be executed according to the execution order. By using the green dependency ports on the left and right sides of the nodes, the execution order of the nodes (Griem et al. 2022) can be freely defined. As can be seen in Figure 1, the *convert-video-to-images* tool node (Figure 3a) is coming next and will be automatically executed. As the name implies, it converts a MP4 video into an image sequence and saves the images in a new folder. This presents the first step for the video analysis of the *in situ* microcompression process. For this purpose, the OpenCV library is used, as can be seen in the python script in Listing 1, Appendix A. The path to the MP4 video to be analyzed serves as a string-input to this node. In our workflow implementation, this path is passed to the node in form of a defined variable. While processing the video within the node, the timestamp of each frame is calculated and used for naming the corresponding image, as indicated in Figure 3b. Later on, the user will be able to clearly see the time at which each deformation of the micropillar takes place.



3.2.3 hld-data-processing node

The 'hld' term in the name of the processing node stands for the .hld output file of the Bruker GmbH nanoindenter. It is a text file format that contains records of time, displacement, force during the deformation and detailed experiment parameters for the indenter setup in the header. The *hld-data-processing* tool node (see Figure 4) is added to extract the maximum loading time of the nanoindenter, where the compression test ends. Bearing in mind that the associated SEM video also records the short period after the indenter is detached from the sample, it is necessary to identify this value and use it to parameterize the *crop-image* node as follows: The maximum loading time will be printed as a standard output (stdout) and piped as an input to the target node (*crop-image* node) through the parameterization ports. The *crop-image* function that is built in this node will extract the time of each image consequently and discard all images that were acquired after the maximum loading time. This helps to avoid unnecessary processing of images that were obtained after the end of the compression test.

Figure 3 (a) *convert-video-to-images* node: converts, for example, the acquired *in situ* SEM video into an image sequence. The timestamp for each image is calculated and embedded inside the name of each image **(b)**.

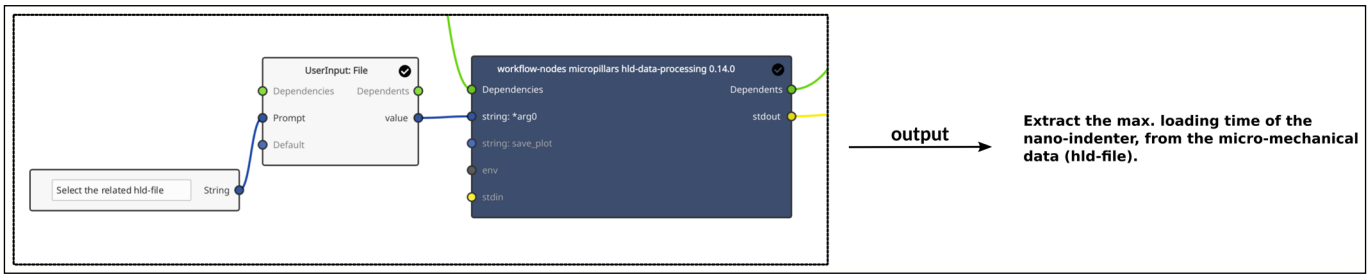


Figure 4 Image of the *hld-data-processing* node, which can be used to, for example, extract the maximum loading time of the nanoindenter.

3.2.4 UserInput: Select Bounding Box node

This built-in node (Figure 5a) is applied here to define the bounding box of the image, which then will be applied in the next tool node. By executing this node, one of the images from the image stack will be visualized inside the KadiStudio workflow execution view. The user will then be prompted to select the desired area of the image (in this case, the area showing mainly the micropillar), which can be done by a simple GUI interaction, as shown in Figure 5b.

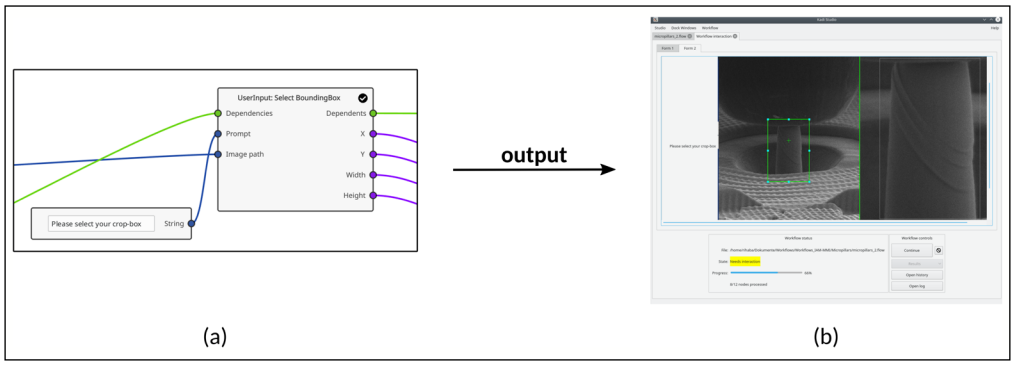
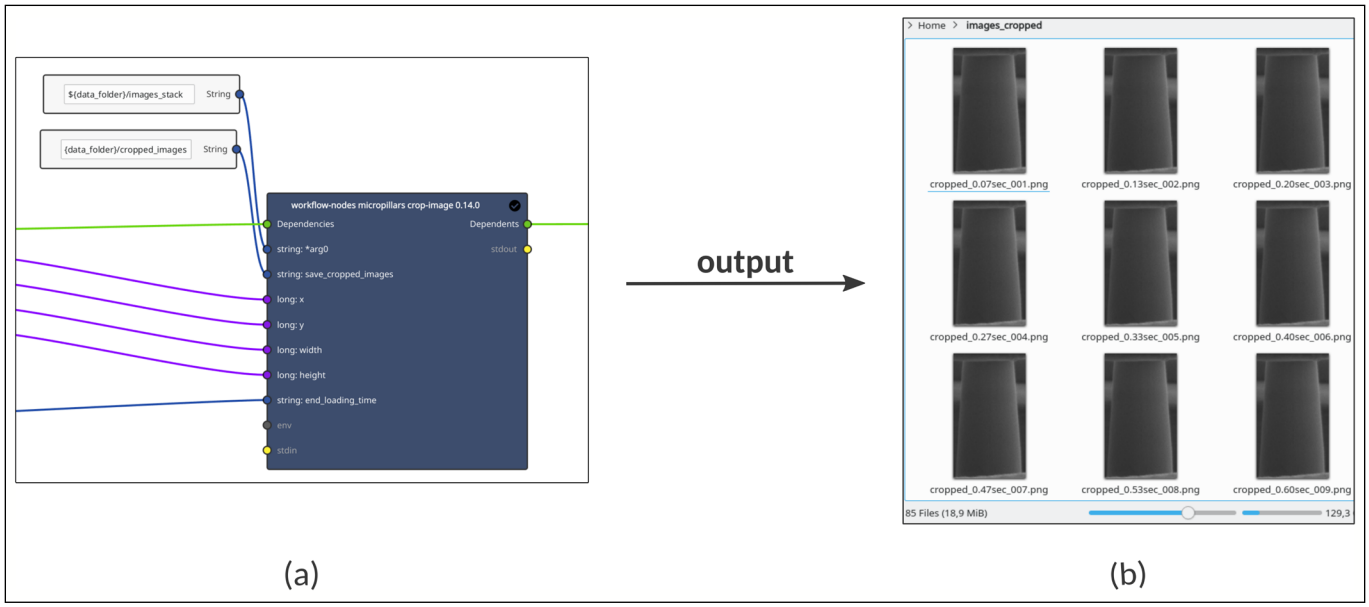


Figure 5 (a) The built-in *UserInput: Select Bounding Box* node. **(b)** The corresponding user interaction GUI created by this node, which prompts the user to select an area of the image. The position and size of the selected area will be available to other nodes in the workflow.



3.2.5 crop-image node

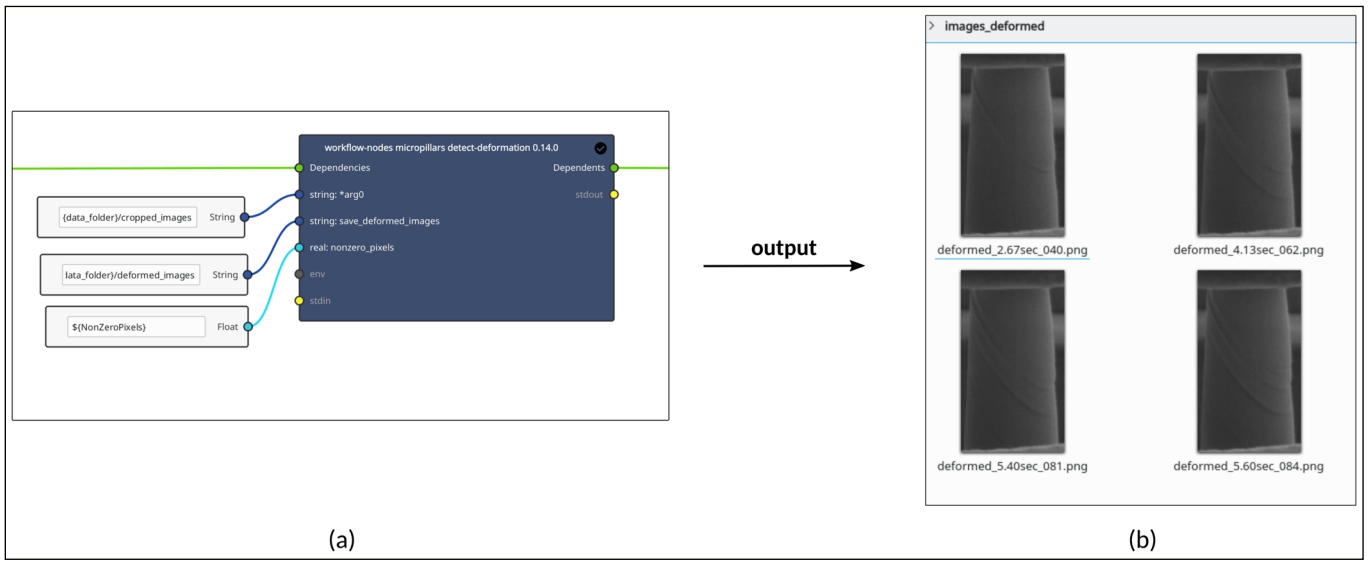
After the user has selected the relevant area of the image, this information is used as an input for the *crop-image* node (Figure 6a), which processes all images in the image stack until the maximum loading time of the nanoindenter is reached. This way, the images are cropped so that only the micropillar appears in each image (Figure 6b).

Figure 6 (a) The *crop-image* node, which takes the user-defined coordinates and size of the bounding box from the previous node (Figure 5a), as an input to crop all images in the image-stack folder, until the maximum loading time of the nanoindenter is reached. This means that only images obtained before or at the maximum loading time will be cropped.

3.2.6 detect-deformation node

At this point, all images are focused on the micropillar present in them; that is, the image pixels now belong mainly to the micropillar structure. This facilitates the comparison of the pixel change in each image in order to automatically detect the shape changes of the pillar: In the *detect-deformation* node (Figure 7a), the paths to the images are saved in a list, and a loop is applied to iterate through this list: it takes every image starting from image 1 and subtracts its pixels from the pixels of the subsequent image, and so on. If the number of nonzero pixels (NZP) counted from the pixel difference between two consecutive images is relatively small, it can be assumed that the micropillar's shape is not changing; that is, there is no noticeable difference between the two images. However, if the surface morphology of the pillar changes due to the increasing applied stress, the pixel change and therefore the number of NZPs will become much higher, eventually exceeding a predefined threshold. In the use-case presented here, the code showed a critical NZP threshold of 5,500 pixels (5.5 kpx). The *detect-deformation* node (Figure 7a) takes this input value and saves all images having a NZP value of ≥5.5 kpx in a separate folder (Figure 7b), and hence it automates the detection of the deformation behavior. We have to mention here that the pixel change due to possible and unavoidable difference in color contrast during the measurement is much lower than that caused by the deformation process and hence does not affect the detection process.

Figure 7 (a) The *detect-deformation* node to automatically detect the images showing micropillar deformation from the cropped images created by the *crop-image node* in Figure 6a. **(b)** The execution result of the node: Images showing micropillar deformation are separately saved in a new folder.



3.2.7 records add-files node

As stated in Griem et al. (2022), the kadi-apy library (Kadi4Mat Team and Contributors 2022a) was created to ensure FAIR documentation of the data generated in a workflow. This library contains CLI tools that can be used as nodes in KadiStudio and offer a link between the workflows and KadiWeb. The access to the repository and its functionalities is ensured via the application programming interface (API) provided by Kadi4Mat (Kadi4Mat Team and Contributors 2022a), which offers a set of defined functions and interfaces to interact with KadiWeb. Accordingly, the *records add-files* node (Figure 8a) is added to the workflow to automatically upload the final output of the workflow, which is in this case all SEM images showing micropillar deformation (with the time values, at which each deformation occurs, embedded in the corresponding image name), to KadiWeb inside a record that is defined by the user (Figure 8b).

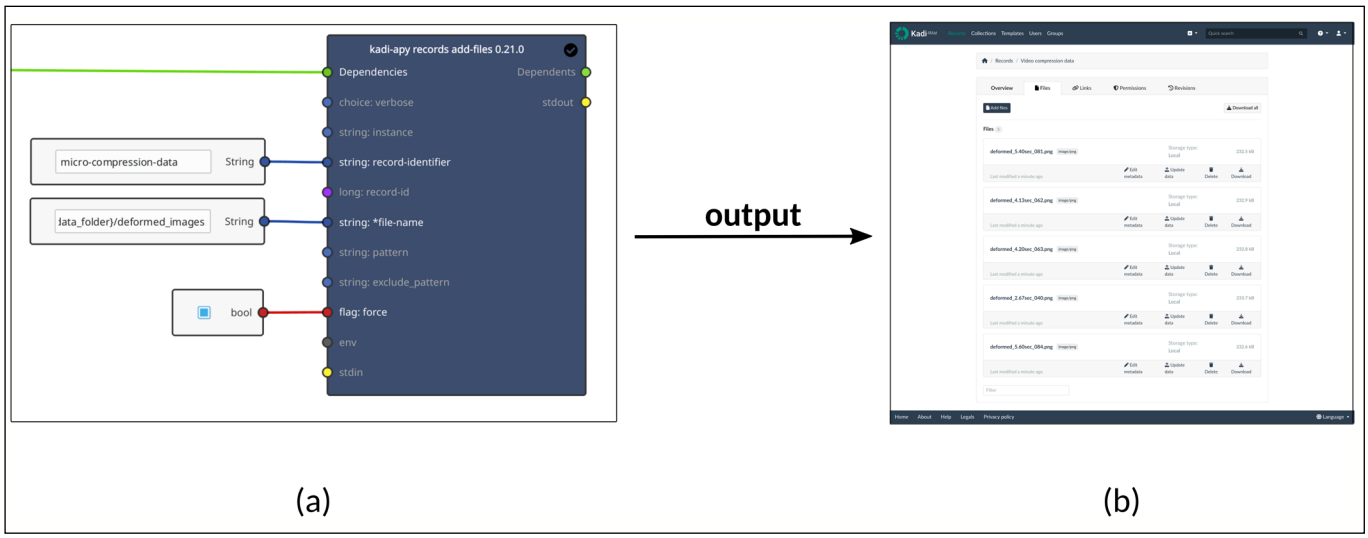


Figure 8 (a) The *records add-files* tool node to upload the data to KadiWeb. **(b)** A screenshot of the record's files overview in KadiWeb showing the uploaded data.

3.3 EFFICIENCY OF THE WORKFLOW IN TERMS OF SPEED AND ACCURACY

To give better insight into the benefits of using the use-case workflow over the manual option, we present in Table 1 a simple comparison between the manual and automated data processing methods of our specific use case.

VIDEO ANALYSIS WAY	SPEED	ACCURACY	REPRODUCIBILITY
Manual	Slow	Medium	Low
Automated workflow	Fast	High*	High

Table 1 A simple comparison between the manual and the automated SEM video processing methods.

* Depends on the provided NZP-value and the SEM-video quality.

For manual analysis of the video, the scientist should (1) run the video and stop it at the point where the micropillar deformation occurs, (2) take a screenshot, (3) open the screenshot image in an image editing program and remove unnecessary parts (crop the image), (4) repeat step 1 to capture other events (further deformation of the micropillar) and repeat steps 2 and 3 accordingly. In the automated analysis method, however, the software takes care of all these steps (in a different way) and at a very high speed compared with the manual option, thus saving the scientist a lot of time. The question of accuracy, for our specific use case, is how accurately the frames in which the deformation appears and the corresponding time can be extracted from the video. In the workflow, the underlying Python code for the custom tool nodes can detect the pixel change between frames and very accurately calculate the time of each frame. In contrast, manual detection depends on the human eye and reaction, which differs from person to person. This in turn affects the reproducibility of the analysis results. Repeated execution of the presented workflow with the same input values will always produce the same output, regardless of who performs it. In contrast, repeated manual analysis of the same use case by different people (and sometimes by the person himself) will rarely produce exactly the same output. Therefore, using automated workflows is much more efficient in terms of accuracy and reproducibility.

4 CONCLUSION

In this work, we present a use-case for the workflow editor of KadiStudio, where the data processing of *in situ* SEM micropillar compression tests is automated. The tests are performed by utilizing *in situ* SEM measurements coupled with a nanoindenter, where the compression behavior of micropillars is tested by simultaneously acquiring different types of data (SEM videos as well as microcompression data: displacement, force, time, etc.). This process is repeated for different micropillars (at least 20) in each sample, and each micropillar will have the same type of dataset. Analyzing these data manually to detect a morphology change in the micropillar and to extract the corresponding time and force values is a time-consuming task. Therefore, automation of such processes by a workflow significantly reduces the workload and hence accelerates the scientific research in general. In addition, it makes the implied image-processing procedure more reproducible because it depends on pixel change and not on human eye. The presented workflow also includes automatic upload of both raw and analyzed data to the KadiWeb repository to ensure FAIR documentation of the analysis process. The kadi-apy library enables a smooth interaction between KadiStudio and KadiWeb. It is noteworthy to mention here that the customized tool nodes are impartible; that is, each node only performs one task. This makes them general in nature and hence easily reusable by other users, ultimately benefiting the whole scientific-community.

APPENDIX A

Listing 1 Tool node example.

```
from xmlhelpy import argument
from xmlhelpy import option

import cv2
import os


# Decoration with @command defines the node type as 'tool'.
@command()

# @argument and @option add new ports to the tool node.
@argument(name= 'video', description= 'Path to the MP4-video.')
@option(name= 'save_images', description= 'The path of the folder,
where the image stack can be saved in')

def convert_video_to_images (video, save_images):
      """convert a video to image-stack"""
    # Create video-capture object:
    cap = cv2.VideoCapture(video)

    try:
        if not os.path.exists(save_images):
            os.makedirs(save_images)
    except OSError:
        print ('Error: Creating directory for images')

    # count the number of frames
    frames = cap.get(cv2.CAP_PROP_FRAME_COUNT)          # total number of frames
    fps = cap.get(cv2.CAP_PROP_FPS)         # number of frames per second

    #loop to save all images in sequence in the created folder (save_images):
    frame_no = 1
    while frame_no <= frames:   # stops when the
    # last frame-number in the stack is reached
        ret, frame = cap.read()
        if ret:
          # calculate the time of the image:
          image_time = frame_no / fps
          name = save_images + f  '/{image_time:.2f}_sec_frame_ {frame_no:03d}.png'
          cv2.imwrite(name, frame)

        frame_no += 1
    print(name)
    # When everything is done, release the capture
    cap.release()
    cv2.destroyAllWindows()
```

ACKNOWLEDGEMENTS

This use-case is partly funded by the Federal Ministry of Education and Research (BMBF) in the project AQua (project number 03XP0315B), the Helmholtz association in the project INNOPOOL MDMC (program No. 43.35.01) with contributions from KNMFi, the BMBF and MWK-BW as part of the Excellence Strategy of the German Federal and State Governments in the project Kadi4X.

C. Kirchlechner and C. Teixeira are grateful for the financial support from the Robert-Bosch-Foundation. Financial support from the Helmholtz Program Materials Systems Engineering: Functionality by Information-Guided Design is gratefully acknowledged by S. Lee, J. Aghassi-Hagmann and C. Kirchlechner. R. Al-Salman thanks Julian Grolig and Nico Brandt (KIT/IAM) for the valuable discussions. We acknowledge support by the KIT-Publication Fund of the Karlsruhe Institute of Technology.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Rihab Al-Salman orcid.org/0009-0003-8257-9932
Institute for Applied Materials (IAM), Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany

Camila Aguiar Teixeira orcid.org/0000-0002-4019-5282
Institute for Applied Materials (IAM), Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany

Philipp Zschumme orcid.org/0000-0002-4821-5719
Institute for Nanotechnology (INT), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

Subin Lee orcid.org/0000-0002-4629-8004
Institute for Applied Materials (IAM), Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany

Lars Griem orcid.org/0000-0002-8093-6356
Institute for Applied Materials (IAM), Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany; Institute for Nanotechnology (INT), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

Jasmin Aghassi-Hagmann orcid.org/0000-0003-0348-041X
Institute for Nanotechnology (INT), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

Christoph Kirchlechner orcid.org/0000-0003-2418-9664
Institute for Applied Materials (IAM), Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany

Michael Selzer orcid.org/0000-0002-9756-646X
Institute for Nanotechnology (INT), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany; Institute for Digital Materials Science (IDM), Karlsruhe University of Applied Sciences, Moltkestraße 30, 76133 Karlsruhe, Germany

REFERENCES

Afgan, E, Baker, D, Batut, B, Van den Beek, M, Bouvier, D, Čech, M, Chilton, J, Clements, D, Coraor, N, Grüning, BA, et al. 2018. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46(W1): W537–W544. DOI: https://doi.org/10.1093/nar/gky379

Deelman, E, Gannon, D, Shields, M and **Taylor, I.** 2009. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5): 528–540. DOI: https://doi.org/10.1016/j.future.2008.06.012

Dehm, G, Jaya, BN, Raghavan, R and **Kirchlechner, C.** 2018. Overview on micro-and nanomechanical testing: New insights in interface plasticity and fracture at small length scales. *Acta Materialia*, 142: 248–282. DOI: https://doi.org/10.1016/j.actamat.2017.06.019

Draxl, C and **Scheffler, M.** 2020. Big data-driven materials science and its FAIR data infrastructure. In: *Handbook of Materials Modeling: Methods: Theory and Modeling*, pp. 49–73. DOI: https://doi.org/10.1007/978-3-319-44677-6_104

Griem, L, Zschumme, P, Laqua, M, Brandt, N, Schoof, E, Altschuh, P and **Selzer, M.** 2022. KadiStudio: FAIR Modelling of Scientific Research Processes. *Data Science Journal*, 21(1): 1–17. DOI: https://doi.org/10.5334/dsj-2022-016

Jain, A, Ong, SP, Chen, W, Medasani, B, Qu, X, Kocher, M, Brafman, M, Petretto, G, Rignanese, G, Hautier, G, et al. 2015. FireWorks: A dynamic workflow system designed for high-throughput applications. *Concurrency and Computation: Practice and Experience*, 27(17): 5037–5059. DOI: https://doi.org/10.1002/cpe.3505

Kadi4Mat Team and Contributors. 2022a. *IAM-CMS/kadi-apy: Kadi4Mat API Library*. Version 0.20.0. DOI: https://doi.org/10.5281/zenodo.5971821

Kadi4Mat Team and Contributors. 2022b. *IAM-CMS/workflow-nodes*. Version 0.13.0. DOI: https://doi.org/10.5281/zenodo.5971840

Kadi4Mat Team and Contributors. 2022c. *IAM-CMS/xmlhelpy*. Version 0.9.2. DOI: https://doi.org/10.5281/zenodo.5971732

Kluyver, T, Ragan-Kelley, B, Pérez, F, Granger, B, Bussonnier, M, Frederic, J, Kelley, K, Hamrick, J, Grout, J, Corlay, S, Ivanov, P, Avila, D, Abdalla, S, Willing, C and **Jupyter Development Team.** 2016. Jupyter Notebooks? a publishing format for reproducible computational workflows. In: Loizides, F and Schmidt, B (eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press. pp. 87–90. DOI: https://doi.org/10.3233/978-1-61499-649-1-87

Liu, J, Pacitti, E, Valduriez, P and **Mattoso, M.** 2015. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, 13: 457–493. DOI: https://doi.org/10.1007/s10723-015-9329-8

Ludäscher, B, Altintas, I, Bowers, S, Cummings, J, Critchlow, T, Deelman, E, De Roure, D, Freire, J, Goble, CA and **Jones, MB,** et al. 2009. Scientific Process Automation and Workflow Management. *Scientific Data Management*, 10(3): 476–508. DOI: https://doi.org/10.1201/9781420069815-c13

Malyar, NV, Springer, H, Wichert, J, Dehm, G, Kirchlechner, C. 2019. Synthesis and mechanical testing of grain boundaries at the micro and sub-micro scale. *Materials Testing*, 61(1): 5–18. DOI: https://doi.org/10.3139/120.111286

Pinaev, et al. 2017. *QtNodes. Node Editor*. https://github.com/paceholder/nodeeditor.

Pizzi, G, Cepellotti, A, Sabatini, R, Marzari, N and **Kozinsky, B.** 2016. "AiiDA: automated interactive infrastructure and database for computational science". *Computational Materials Science*, 111: 218–230. DOI: https://doi.org/10.1016/j.commatsci.2015.09.013

Uchic, MD, Dimiduk, DM, Florando, JN and **Nix, WD.** 2004. Sample dimensions influence strength and crystal plasticity. In: *Science*, 305(5686): pp. 986–989. DOI: https://doi.org/10.1126/science.1098993

TO CITE THIS ARTICLE:
Al-Salman, R, Teixeira, CA, Zschumme, P, Lee, S, Griem, L, Aghassi-Hagmann, J, Kirchlechner, C and Selzer, M. 2023. KadiStudio Use-Case Workflow: Automation of Data Processing for *in Situ* Micropillar Compression Tests. *Data Science Journal*, 22: 21, pp. 1–11. DOI: https://doi.org/10.5334/dsj-2023-021

Submitted: 30 March 2023
Accepted: 25 May 2023
Published: 10 July 2023

Data Science Journal is a peer-reviewed open access journal published by Ubiquity Press.