
Article:
Bonney, M.S., de Angelis, M., Dal Borgo, M. et al. (4 more authors) (2022) Development of a digital twin operational platform using Python Flask. Data-Centric Engineering, 3. e1. ISSN 2632-6736

https://doi.org/10.1017/dce.2022.1

Data-Centric Engineering (2022), 3: e1
doi:10.1017/dce.2022.1

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Development of a digital twin operational platform using Python Flask

Matthew S. Bonney[1], Marco de Angelis[2], Mattia Dal Borgo[3,4], Luis Andrade[5], Sandor Beregi[6], Nidhal Jamia[7] and David J. Wagg[1,*]

[1]Dynamics Research Group, Department of Mechanical Engineering, University of Sheffield, Sheffield, United Kingdom
[2]Institute for Risk and Uncertainty, University of Liverpool, Liverpool, United Kingdom
[3]Department of Mechanical Engineering, KU Leuven, Leuven, Belgium
[4]DMMS Lab, Flanders Make, Leuven, Belgium
[5]Department of Engineering, University of Cambridge, Cambridge, United Kingdom
[6]Department of Engineering Mathematics, University of Bristol, Bristol, United Kingdom
[7]Faculty of Science and Engineering, Swansea University, Swansea, United Kingdom
*Corresponding author. E-mail: david.wagg@sheffield.ac.uk

Received: 28 July 2021; **Revised:** 15 November 2021; **Accepted:** 07 January 2022

Keywords: Digital twin; Flask; operational platform; open source; Python

Abstract

The digital twin concept has developed as a method for extracting value from data, and is being developed as a new technique for the design and asset management of high-value engineering systems such as aircraft, energy generating plant, and wind turbines. In terms of implementation, many proprietary digital twin software solutions have been marketed in this domain. In contrast, this paper describes a recently released open-source software framework for digital twins, which provides a browser-based operational platform using Python and Flask. The new platform is intended to maximize connectivity between users and data obtained from the physical twin. This paper describes how this type of *digital twin operational platform* (DTOP) can be used to connect the physical twin and other Internet-of-Things devices to both users and cloud computing services. The current release of the software—DTOP-Cristallo—uses the example of a three-storey structure as the engineering asset to be managed. Within DTOP-Cristallo, specific engineering software tools have been developed for use in the digital twin, and these are used to demonstrate the concept. At this stage, the framework presented is a prototype. However, the potential for open-source digital twin software using network connectivity is a very large area for future research and development.

Impact Statement

Digital twin technology has been recognized as an important new method for a wide range of industries, and other sectors, to obtain significant additional value from data. In order to be implemented in practice, a digital twin needs an operational platform. This paper presents new results on these operational platforms, including associated open-source code for a demonstrator application using Python and Flask.

1. Introduction

The digital twin concept has been widely studied in recent years, as described in the recent review papers: Fuller et al. (2020), Jones et al. (2020), Liu et al. (2020), Minerva et al. (2020), Wagg et al. (2020), and Niederer et al. (2021). In the context of engineering applications, a digital twin has four main elements: (a) models (both physics- and data-based), (b) data, (c) digital connectivity, and (d) knowledge (both contextual and expert; Gardner et al., 2020). In order to realize a digital twin in practice, an *operational platform* is required. This will need to incorporate the necessary software and hardware components that enable the digital twin to interact with the physical twin that it relates to. In this paper, we will refer to such a combination of software and hardware as a *digital twin operational platform* (DTOP). The idea of a DTOP is closely related to that of a digital twin information system, which is a concept that has been developed in the construction industry (Sacks et al., 2020).

As pointed out in the recent review by Minerva et al. (2020), the digital twin concept is intrinsically linked to the development of Internet-of-Things (IoT) applications (also in this context sometimes called the industrial IoT). There are several reasons for this, but from the point of view of creating an operational platform, the requirement for connectivity between the digital twin and the physical twin is the most relevant to the current work, with a secondary focus on the connectivity between the operational platform and third-party software such as (in an engineering context) finite-element solvers.

The idea of the IoT is entirely based on providing connectivity for physical devices via the Internet, primarily as a method for accessing and transferring data, but also in some cases to send control and command signals to the device as well (Gilchrist, 2016). Therefore, it is natural to think of DTOPs having an IoT network containing the relevant physical twin and other hardware elements as a starting point. Specifically, we assume that (a) the physical twin(s) is (are) connected to the Internet (or a local-area network, LAN) and that (b) data from sensors attached to physical twin can be continuously communicated across the network (in other words, that data acquisition (DAQ), processing, and storage are already provided). With these two requirements met, a DTOP can be constructed based on the addition of other key components such as a user interface and data processing capabilities. As noted already, in many digital twin systems, there is a requirement to output control signals and other commands to the physical twin. This is also possible in the proposed DTOP framework, and an example is shown in Section A.1 in the Online Appendix (see also Gardner et al., 2020).

In terms of digital twin software developments to date, most activity has been driven by proprietary software vendors (see the review in Minerva et al., 2020), resulting primarily in closed-source products. However, there are major benefits by enabling interoperability between different parts of the digital twin and even between multiple different twins. Therefore, an objective of the research community is to create greater openness in platforms and associated software being used, and this is already being pursued in the area of IoT (see, e.g., Platenius-Mohr et al., 2020). While the majority of digital twin solutions are closed-source, there are only a few open-source frameworks available. The largest framework is Eclipse Ditto from the Eclipse foundation (Eclipse Ditto, 2021). The main purpose of Ditto is the connection IoT devices to data storage and provide an Internet-based interface for real-time operation. This, however, does not focus on the nonoperational calculations that are typically done in complex engineering systems asset management (such as failure prediction and uncertainty quantification); thus, Ditto becomes more rigid to implement modifications that are required for these engineering systems.

In this paper, this ethos is applied to the development of an open software that consists of a prototype browser-based DTOP called DTOP-Cristallo. The platform has been developed using the Python/Flask framework in order to provide a user interface via web pages. This format should maximize accessibility for as many users as possible. It also allows a direct route for connectivity to the physical twin and other cloud-based services that may be required. However, there are trade-offs to be made. For example, cyber security of the digital twin is much more difficult to manage using a web-based platform. Using a web-based platform opens the possibility of using application program interfaces as a method of obtaining interoperability (Scheibmeir and Malaiya, 2019). As will be seen in this paper, the proposed DTOP tries wherever possible to make use of this approach.

This paper is structured as follows: Section 2 describes the overall framework from which a DTOP can be built, and details of the key requirements for a DTOP. To demonstrate this concept of an operational platform, details of DTOP-Cristallo are presented in Section 3, and in particular, some possible use cases for a DTOP are highlighted. Section 4 discusses implementing the DTOP framework as a server-based DTOP that allows for great accessibility that promotes wider collaborative activity and access to data. Concluding remarks are given in Section 5.

2. Connectivity and Accessibility Formats for Digital Twin Operational Platforms

The aim of a DTOP is to enable users to manage the physical twin using the specific capabilities of the digital twin. When there are multiple users, then accessibility becomes an important part of the DTOP. More specifically, accessibility refers to who and where a user can access the digital twin, and within the twin which data and other information each user can have access to. As this paper is presenting a prototype system, the current focus is on the "where a user can access" rather than "who can access." Providing different levels of access to users via Flask is something not covered in this paper, but can be provided, as described, for example, by Grinberg (2018). To ensure consistent terminology, this paper refers to accessibility as the methods by which a user can access the digital twin via the DTOP. Connectivity is considered to mean how the digital twin is connected to other data sources/simulations. Both connectivity and accessibility options are design choices made based on the system of interest, the expected user base, required simulations, and other various aspects that are realized through the DTOP.

With regard to the programming for DTOP-Cristallo (demonstrated in Section 3), the choice was made to use Python as the base programming language with Flask used to provide the browser-based interface. These choices were made due to the universality and open-source nature of Python and Flask. Flask is a Python-based module that produces a web page-like application that utilizes HTML, CSS, and JavaScript (see, e.g., Grinberg, 2018; Nixon, 2014). An additional benefit for using Flask is that this work was undertaken as part of a multi-institutional research project (see the Acknowledgment section for details; published work from the project includes Wagg et al., 2020; Balatti et al., 2021; Beregi et al., 2021; Chakraborty and Adhikari, 2021; Chatterjee et al., 2021; Jamia et al., 2021; Gray et al., 2022), and this format was a natural choice for multiuser access with users based at multiple different geographical locations.

Digital twins are context-dependent (see, e.g., Jans-Singh et al., 2020; Wagg et al., 2020; Kapteyn et al., 2021; Niederer et al., 2021), meaning that they should reflect the specific physical twin that they relate to. That said, the accessibility of a DTOP has some generic formats that can be broadly grouped into three categories, as shown in Figure 1. The first accessibility method is through a standalone framework (top panel in Figure 1). This framework utilizes a single access point to the digital twin, typically through an executable. Data are acquired directly to the standalone machine, and all operations are carried out on this one computer. Using this framework provides the maximum amount of data security, but greatly limits user access, particularly remote access capabilities. This format is typically created by private generators of digital twins in order to maintain proprietary information, and is particularly applicable to systems that requires very high levels of security, such as military applications or management of critical infrastructure, with the limitation that all functionality must be provided within the standalone system.

The next accessibility format for a DTOP involves the use of a LAN (central panel in Figure 1). This method allows access to multiple users; however, they are limited to being on the same local network. This format can be especially useful for multimachine digital twins, such as a process digital twin, or automated manufacturing equipment such as robotic welding. This allows for an operator to monitor the equipment in a safe location, but also allows a technician to inspect the system for repairs. The LAN-based framework allows for other computational infrastructure, such as local high-performance computing (HPC) to be used, if it is available at the location. This methodology allows for more user access to the digital twin while maintaining the data security of all operations at the specific location, but there is limited access to data and other services available on wider networks.

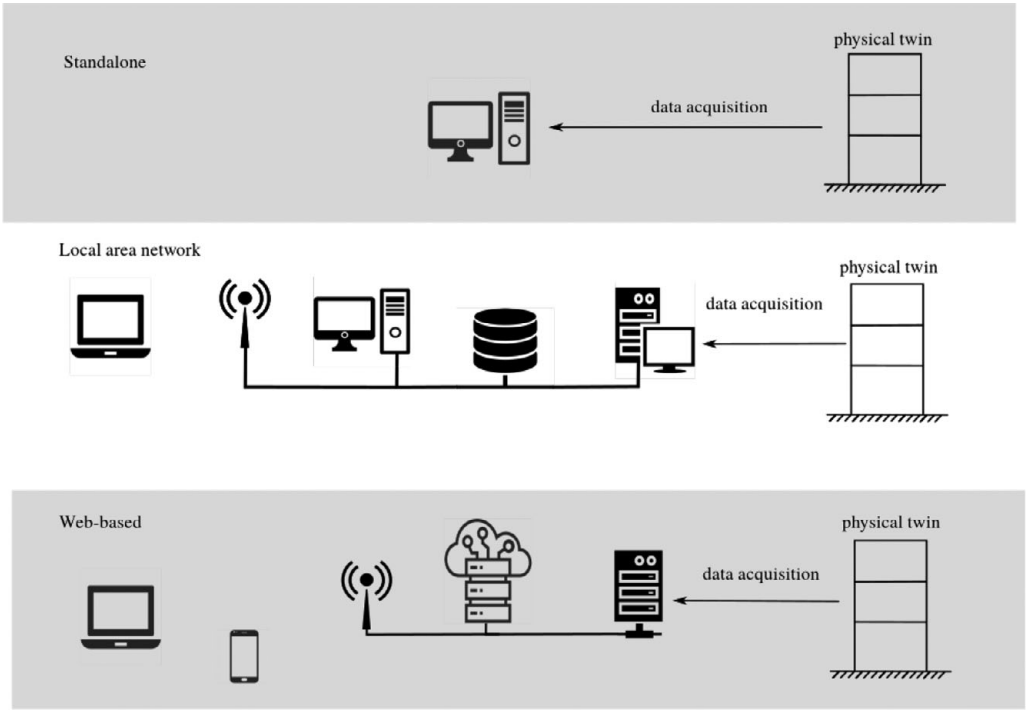

Figure 1. *The three proposed methods of accessing a physical twin via a digital twin operational platform.*

The last accessibility format is the web-based (bottom panel in Figure 1; also called server-based) DTOP. This offers the most wide-ranging connectivity and access to data and services, and it can enable multiple users to interact with the physical twin to carry out asset management or other tasks after the structure is deployed. This framework allows access to approved users through any device that is connected to the Internet. While this requires a much larger effort to maintain data security, it also facilitates multiple users to have access, and can be used to promote additional collaborations if appropriate.

The appropriate accessibility format will depend on the application. However, as a key benefit of a digital twin is the use of big data, and data modelling techniques such as machine learning, using a DTOP in a standalone format (similar to the digital twins provided by private vendors) is highly constraining. Furthermore, for many engineering systems, such as large structures (and the related construction industry; Sacks et al., 2020), offshore facilities or unmanned air vehicles (Kapteyn et al., 2021), human access to the system ranges from difficult to impossible. Therefore, remote user access across the Internet is often a highly beneficial format.

The browser-based operational platform format can be broken down into three "layers," as shown in Figure 2. The foundation layer is the IoT layer, where physical twins and other devices are connected to the network using bespoke hardware. This also includes local DAQ and control hardware, sensors and actuators. The interface layer provides access to users via a web server that coordinates and schedules the required tasks within the workflow. This layer uses the network to connect the user to the data provided by the IoT layer and cloud-based services. The services provided in the cloud computing layer would typically include data storage, HPC, and any other remote computing facilities required as part of utilizing the digital twin.

For DTOP-Cristallo, prototypes of both the standalone format and server-based format have been developed. The standalone version has been developed as a demonstrator that can be easily downloaded and run by users on a single machine (see the details in the Data Availability Statement section) to

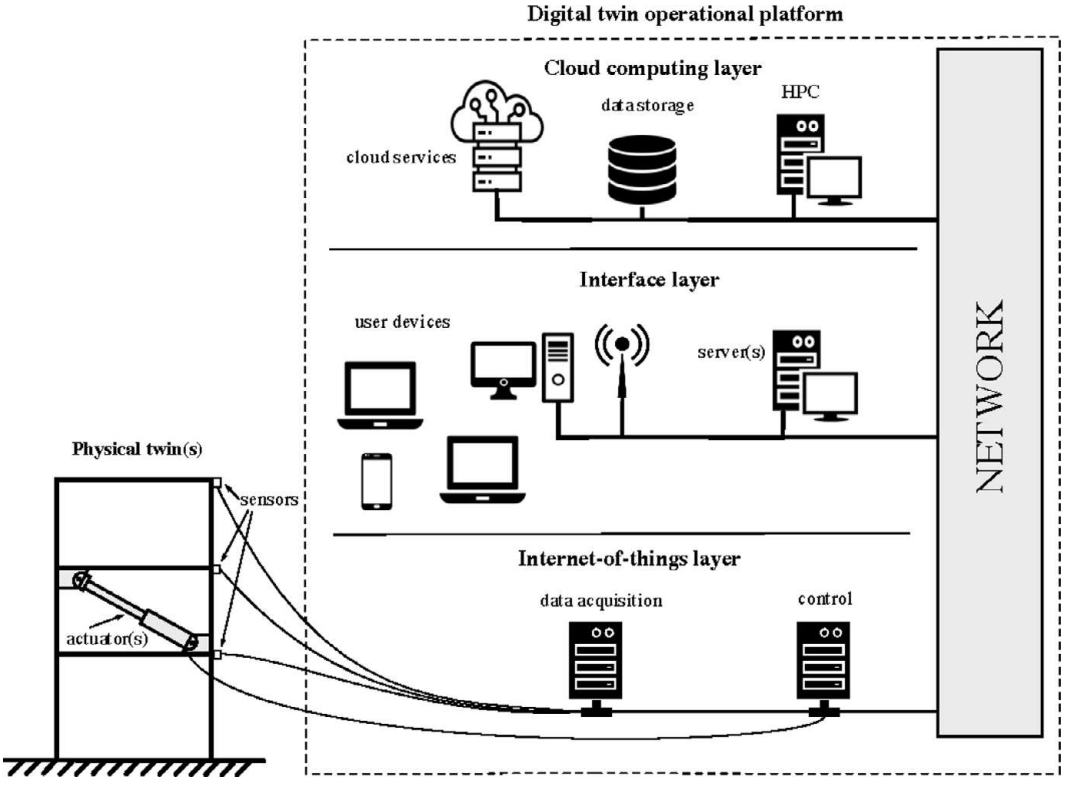

Figure 2. *A schematic representation showing the three "layers" of a browser-based digital twin operational platform (DTOP). Note that (a) the sensors and the actuators are part of the IoT layer, but shown here outside the DTOP dashed box for convenience, and (b) the actuator system shown in the IoT layer is for conceptual insight only, and it is not supposed to represent a typical actuator configuration in a three-storey structure.*

demonstrate the development of the interface layer. The server version is being used as part of an ongoing research project, and helps facilitate the geographically distributed nature of the partners in the project. Specifically, using the server-based framework allows researchers from multiple universities and multiple testing locations to contribute to the code for the DTOP. As a result, DTOP-Cristallo provides a demonstrator standalone version and a proof of concept for the web-based DTOP.

2.1. Outlined requirements

To aid in the discussion of operational platforms, several key requirements are now described. While these requirements need not be jointly satisfied for every applications, they include the ability to:

1. Provide access to appropriate users and maintain data security. This includes authentication and logins.
2. Monitor the current state of the physical twin. This can include either a direct stream of sensor information or an accident report/periodic type of setup.
3. Make modifications to various aspects of the physical twin. Examples would include modifying any active control parameters and initiating safety protocols.
4. Perform/schedule simulations to aid in decision-making. This can include a wide variety of simulations such as failure analysis, electrical storage requirements, and acoustic levels for passenger comfort.

5. Centralize results from both the physical and digital twins to aid in decision-making, including the use of databases with proper storage redundancy.
6. Evolve digital twin parameters over the life of the system to ensure accurate comparability between the physical and digital twins. This evolution can include aspects such as material degradation, reported repairs, and incidental damages.

While an ideal DTOP may incorporate all of these components, this paper presents just a limited set of requirements in order to demonstrate the DTOP concept. Specifically, for DTOP-Cristallo, the focus is mainly on the aspects of simulations for a digital twin, primarily items 1 and 4. For some of the other requirements in the list, such as items 2 and 3, there are constraints due to the specific application. This is because, during the development of DTOP-Cristallo, the physical twin corresponds to scaled three-storey structure experimental rigs in the Universities of Bristol, Sheffield, Southampton, and Swansea Laboratories—an example of one of these structures is shown in Section A.6 in the Online Appendix. However, these experimental rigs are not continuously connected to the digital twin, and therefore cannot be physical twins in the strictest sense. Instead, connection with the physical asset can be enabled to carry out specific testing. However, this type of testing requires specific local DAQ hardware (and control if appropriate), and therefore cannot be easily provided in an open software project such as DTOP-Cristallo. Instead, for the purposes of demonstrating the idea, the data are either prerecorded or simulated.

2.2. Methodology for web development with Python and Flask

Following the general definition of the envisaged DTOP, this section briefly outlines how the implementation of DTOP-Cristallo fits within the general open-source framework. The language at the core of the DTOP-Cristallo is Python, which is licensed under the *PFS license agreement.* In addition to the PFS license, starting from Python 3.8.6, examples, recipes, and other code in the documentation are dual licensed under the PSF and the *Zero-Clause BSD license.* The choice of Python as the core language does not need much justification, as it is to date the dominant programming language in a variety of web-based and scientific applications. Despite the criticism about its speed, Python can also be used to run expensive algorithms thanks to the widely available and accessible libraries for vectorized computing such as Numpy, Redis, and CUDA.

Within the Python ecosystem, Flask stands out as a server tool for a variety of reasons. First, Flask is small with only a few lines of source code, is released under the *BSD 2 license agreement*, and has no required framework for accessing databases, validating web forms, authenticating users, or other high-level tasks. In other words, Flask gives the developers the flexibility to be creative and enable their framework to be problem-specific. This is in contrast with other server tools, where most choices are hard to change or adapt to specific needs. Flask has three main dependencies:

- routing, debugging, and web server gateway interface;
- Jinja2 templates; and
- command-line integration with Click.

Because these dependencies come with the installation of Flask, the only requirement to run Flask is a computer with Python installed. Flask has also been chosen because of its accessibility options and its thorough documentation, exemplified by the comprehensive blog of Miguel Grinberg (Grinberg, 2018).

Flask can be deployed in all three of the methods described in Figure 1. For the LAN and web-based versions, Flask allows for deployment based on IP, thus allowing for remote hosting and access. Flask provides the flexibility to structure the code in a way that the web-interface design is reasonably independent from the development of the underlying mathematical code. This underlying mathematical code includes both the code for dispatching the information gathered from the user, recorded data, and the pure mathematical simulations, which can also include the utilization of cloud-based tools such as HPC. The design of the web interface can be done using the popular triad HTML5, CSS, and JavaScript for

responsive, interactive, and animated browser-based graphical interfaces. For a more in-depth explanation of how a new tool can be contributed to DTOP-Cristallo or how to implement this framework for a new system, the online documentation (available in the documentation tab within Cristallo) gives a detailed explanation of this process.

The open-source Plotly graphing libraries, under the MIT license, are used to make the interactive plots. There are also working examples of Flask connecting with other popular tools for graphical interfacing, like React.js, which opens up exciting possibilities for nonbrowser-based applications. With Flask, the popular dichotomy frontend/backend can be fully exercised with only a few lines of code. Using Flask, the power of the high-level Python programming language can be fully harnessed, providing the interconnectivity for multiple microcomponents that range from scientific computing to data management up to graphical interfacing. With this setup, the broad user base can use an Internet browser to interact with multiple components of the digital twin and trigger different computations, without any particular programming knowledge, while the knowledgeable user can download the open-source project and make customized changes to suit their required purpose. It is hoped that this open-source approach will contribute to making digital twin technology more widely and easily accessible within the engineering community.

3. DTOP-Cristallo

In DTOP-Cristallo, only four of the six requirements from Section 2.1 are directly addressed (namely 1, 4, 5, and 6). The two that are not addressed are related to the IoT layer of a DTOP, which will be investigated as part of further development work. To demonstrate the methodology on a three-storey structure, DTOP-Cristallo was developed. This system is a simple example structure, while chosen not to over complicate construction at a range of locations for experimental testing, mimicking prototype testing. Furthermore, keeping the physical twin structure relatively simple allows for easier verification of results to known theoretical values. The intention is that once the framework is verified on a relatively simple physical twin, then it can be applied to more complex applications.

When a user connects to DTOP-Cristallo, the landing (also home or front) page of the system is shown in Figure 3. The home page includes an animated GIF of the 3D CAD model of the physical twin. The full 3D CAD model is available via the "CAD Model" tab on the left-hand side. This model is generated using AutoDesk Fusion 360, which is a cloud-based drafting tool by Autodesk that can generate an embedded html script to visualize the system.

Figure 3 also shows the simulation suite (tool kit) available to run on this system, for example, the icons and labels listed on the right-hand side of the page in Figure 3. Each of these tools has been developed as a bespoke piece of Python code to carry out a specific task or set of tasks. These tools are also available on the left border by highlighting to corresponding icon. Upon hover, these icons expand to give the name of the tool for simplicity. These tools are not designed to cover the whole spectrum of simulations performed on digital twins, but instead to demonstrate some of the potential simulations that can be achieved. Currently, only a selection of physics-based models or prerecorded experimental results are incorporated. However, they are not limited to this, and it is possible to write additional Python tools to carry out any specific task required (as an open-source software, anyone can do this in principle). To better explain the potential utility of DTOP-Cristallo in a research context, the remainder of this section describes the three categories that these tools fall under. Figure 4 shows the interaction of these three categories with the other components within the operational platform. More detailed discussion of each tool individually is given in the Online Appendix.

3.1. Self-developed simulations

The first category of tools demonstrated in DTOP-Cristallo consist of simulations that are developed in Python by the researchers. In DTOP-Cristallo, these tools include the structural vibration control (Section A.1 in the Online Appendix), design under uncertainty (Section A.2 in the Online Appendix),

Figure 3. *Landing page of DTOP-Cristallo showing the available simulations and a CAD model of the three-storey structure.*

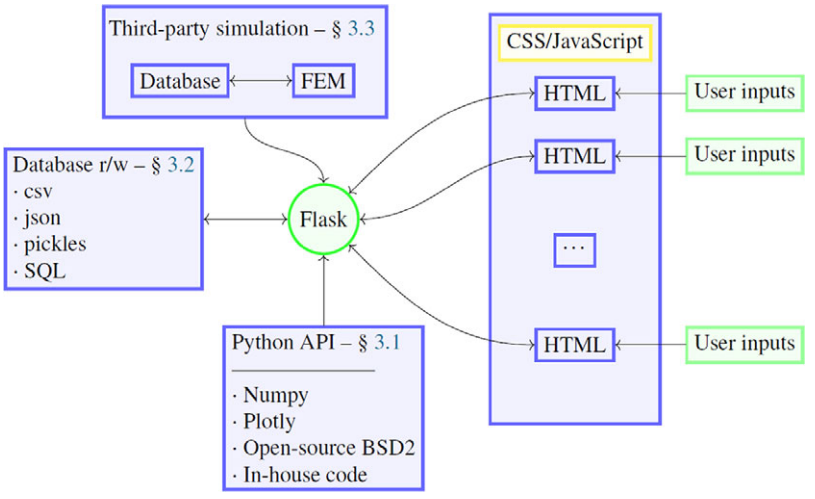

Figure 4. *Diagram of interactions between components and the main categories of tools in the DTOP-Cristallo. The arrow directions signal the flow of data from components and subcomponents.*

and uncertainty propagation (Section A.3 in the Online Appendix). These tools are primarily comprised of two components: the front end that converts the user inputs (including float, Boolean, and string values) into Python variables, and the back end that is a functional calculation that performs the desired

computation. The outputs from these calculations are then presented to the user through graphs and text shown on the browser interface.

This category of tools was the original type of simulations that was envisioned for this DTOP framework. For a typical complex engineering system, there tends to be a large number of simulations that are not available in commercial software, so they are developed via custom programming. Historically for aerospace structures, for example, this was done typically in FORTRAN due to the capabilities at that time. Currently, most of these simulations are being converted or redeveloped in modern programming architecture such as Python. This is also beneficial for engineering research. Due to the ease of use, free license, powerful capabilities, and wide applicability, Python is a perfect computational architecture for developing novel simulations to advance engineering technology. The users can also plug in their own self-developed code libraries, for example, for sensitivity studies, uncertainty quantification, and so on, provided that they are written in or callable from Python.

3.2. Database read/write

While the Python-based tools mentioned in Section 3.1 are useful for simulations that only require user-submitted parameters, there are a large amount of simulations that require external information such as experimental results. To expand on the possible simulations that can be performed, the second category of tools implemented in DTOP-Cristallo utilize information that is stored in database files with the current implementation being limited to local databases such as Comma-separated Values (CSV) files. For DTOP-Cristallo, there are two database tools implemented that include nonlinear control-based continuation (Section A.4 in the Online Appendix) and experimental data cross-validation (Section A.6 in the Online Appendix).

These two tools utilize files/data in two very different approaches. For the nonlinear control-based continuation, the database interaction is used to store simulation results performed during the parameter sweeps (both frequency and excitation levels). Taking the frequency sweep, for example, once the calculation is performed for the excitation frequency, the data are stored in a file before a new excitation frequency is calculated. This is done in order to reduce the required memory in order to store this information, thus putting less hardware requirements on the machine that is performing the calculations. Despite the reduction of hardware requirements, there is a computational efficiency loss through using the read/write operations limited to both the memory and hard-drive storage speed. The other tool, the experimental data cross-validation, uses the database operations in a more straightforward approach. This tool uses the experimental and calibrated model frequency response functions that are recorded in formatted files. While this current implementation only displays the comparison of various prototypes/models to the user, this framework enables the possibility of a more complicated simulation, such as using the experimental data to calibrate/update the finite-element model.

One issue that arises from this category relates to data security and transfer between the physical and digital twins, as well as between users. In the current implementation, these files are managed via the Git repository as is the rest of the code. However, Git has limitations especially when more data are collected and stored. This is a large issue when considering the entire life span of a deployed system. This leads to the use of a remote database, such as an SQL database, that maintains data security and integrity such that the DAQ has an easy path to storing data. This is discussed more detail in Section 4.1.

3.3. Third-party simulation software

The final category of tools implemented into DTOP-Cristallo involves the utilization of licensed or open-sourced third-party software in order to perform simulations. Aside from novel simulations that are custom programmed, there are many quantities of interest that are classically known and implemented into commercially available software. There are many examples of these interests such as creating the finite-element mesh, calculating natural frequencies, stress analysis, multiphysics-based contact, and electrical wiring configurations. For DTOP-Cristallo, only one tool uses a third-party software, which is

the finite-element analysis (FEA) model generation (Section A.5 in the Online Appendix) using ABAQUS.

The FEA tool takes in the user input relating to two main aspects, the mesh refinement and the material properties, and performs a frequency-based analysis to get the natural frequencies. This is done through two main steps: first one is the use of verbose code. ABAQUS CAE is based on the Python programming language that allows Python code to be used to generate FEA models. In DTOP-Cristallo, the majority of this code is fixed (geometric properties, element types, etc.) and is stored in a text file. This file is generated by the designer either through expert knowledge in the implementation or through the macrorecorder option available with some small variations to account for variable inputs given by the user. To make a modification to a new simulation (such as a stress analysis or substructure generation), the user needs to make some changes to this text file, mainly changing the "step" and "load" aspects of the script. Second, the user input is taken and combined with the fixed code to create a singular Python script. After these two steps, then ABAQUS is called to run the Python script via the command prompt called within Python. Currently, DTOP-Cristallo only calls ABAQUS to perform the simulation and does not display the results to the user via postprocessing. This is an area of ongoing work, but currently, the user has to manually open the output files to gather the information.

In terms of using FEA programs in general, there are both commercial and open-source options available. The decision to use a commercial program was chosen based on the applicability to industrial systems. Complex engineering system designers/operators have a large degree of trust in the commercially available programs compared to open-source programs. This trust is an important part of dealing with modern complex engineering systems. So, in order to build the robustness of this framework, DTOP-Cristallo implements a first option for connecting the DTOP framework to third-party software. Instead of convincing the engineering designers/operators to accept the open-source software, the DTOP framework works on the centralization of simulations and computational resources, which can utilize queues that are discussed more in Section 4.1.

3.4. Git repository for standalone version

To aid in communication and demonstration, a standalone version of DTOP-Cristallo is available to download via a Git repository (see the Data Availability Statement section). To install the DTOP-Cristallo standalone version, please visit https://github.com/Digital-Twin-Operational-Platform/Cristallo and follow the quick-start guide available in the README.md file, which can be found at the GitHub landing page. The standalone version is fully functional for the tools discussed in this section. However, the upgrades that will be presented in Section 4 are not implemented into the standalone version due to the nature of the upgrades. The server-based version is not publicly available due to restrictions in hosting and security, but is predominantly used as a test bed for accessibility and connectivity features such as distributed computing and database storage.

While the implementation of the DTOP framework is demonstrated on DTOP-Cristallo, this framework is not limited to the tools described in the Online Appendix or the three-storey structure. One advantage of this framework is the ability to easily add in new tools/simulations based on the needs of the digital twin. This addition only requires the generation of an HTML file(s) and the scientific code to perform the simulations. Incorporating the DTOP framework to a new system, however, requires more attention since each tool must be redesigned. This new DTOP can utilize the work demonstrated in DTOP-Cristallo or can be redesigned to better fit the requirements of the digital twin.

4. Server-Based Upgrades and Benefits

For DTOP-Cristallo, there are multiple versions of accessibility thanks to the implementation using Flask, with the two main versions being the standalone version and the server-based version. The standalone version is the version that is downloadable via the Git repository discussed in Section 3.4, and the server-based version is a version of the DTOP that is hosted online that is available on any

device with Internet access. The server-based version is not currently publicly available since it is used as a demonstrator within the DigiTwin project. While the tool calculations are identical, the accessibility differences allow for upgrades of aspects like ease of use and remote computations. For this server-based version, DTOP-Cristallo utilizes an SQL database with a PostgreSQL back end (Momjian, 2001) and a Redis server queue for scheduling and distributed computing (Copeland, 2008; Macedo and Oliveira, 2011).

4.1. Upgrades implemented in the server-based version

PostgreSQL is a type of relational database that is able to store arrays as a single entry. Other SQL databases do not have this capability, so PostgreSQL is chosen to better store the expected datasets for digital twins. In addition, in order to aid the engineers developing the DTOP in Python/Flask, a module called SQLAlchemy is used to access the database directly from Python (Copeland, 2008; Myers and Copeland, 2015). This allows a shared programming language, reducing the complexity for developing new tools within the DTOP.

The other main upgrade for the server-based version is the Redis server queue. Redis is an in-memory data storage cache used for scheduling numerical simulations. It is used to manage simulation queues and distribute simulations to various workers for performing the calculations via distributed computing. Using Redis in a DTOP provides two main services. The first service is the ability to schedule multiple simulations. There are many typical analyses that require the use of multiple simulations, such as sensitivity studies and uncertainty propagation. In addition to the time management efficiency increase, using Redis also allows for multiple analysts to work at the same time using the same computational resources such as a high-performance computers. This is particularly useful for large projects where a team of analysts are working on the same system, but different aspects, such as structural response and electrical properties, for example. In addition to the scheduling, Redis also enables the ability to use distributed computing. The scheduler is able to assign the simulations in the queue to any available worker. This allows for multiple machines to be used for performing various simulations. These machines have no hardware requirements, so they can be high-performance computers, standard desktops, or virtual machines depending on the budget and capabilities available.

The logic of the server-based DTOP system is shown in Figure 5. Here, the connectivities between different parts of the server workflow are shown with arrows, and boxes are used to indicate the possibility of separate machines. In general, there are at least five machines needed to perform numerical simulations and six to incorporate physical testing. While it is possible to have each component be separate hardware, it is also easy to combine some of the components together, such as the Redis server and the Flask machine. In addition to combining components into a single machine, it is also easy to incorporate cloud computing to create virtual hardware to work as part of the DTOP.

For DTOP-Cristallo's server-based version, each of these machines is cloud-based except for the DAQ machine(s). These are the computers that are connected to the physical twin that reads and records data

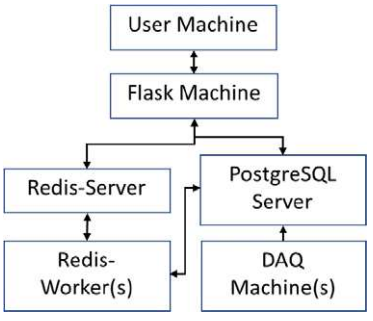

Figure 5. *Example of the implemented server layout.*

from the attached sensors, for example, the IoT layer of Figure 2. In the current implementation, these DAQ machines only interact with the PostgreSQL database, but in future implementations, it is planned that they will also interact with the Flask machine in order to address the remaining two DTOP requirements (items 2 and 3 listed above in Section 2.1). This multimachine procedure has been tested using solely physical hardware, purely virtual hardware, and the combination of both physical and virtual hardware. All combinations of the tested hardware worked as expected. The transition between physical and virtual hardware required very little programmatic changes, with most occurring on the IP address/URL for the informational transfer between machines, particularly to ensure that the Redis server and PostgreSQL server are correctly linked to the workers and Flask machines.

5. Conclusions

Digital twins are a very important development in system design and maintenance for a large number of industries. However, there has been little, if any, research focused on the accessibility and connectivity between the digital twin, the physical twin, and the user. This paper has examined the possible formats for connectivity, and put forward broad definitions that are applicable to real engineering systems, in particular in the context of DTOPs. In general, there are three accessibility categories: (a) standalone, (b) LAN-based, and (c) web-based (or server-based). These three formats describe the interaction between the user and the digital twin with current work being performed to connect the user to the physical twin. The selection of the accessibility format is highly related to the context of the physical twin, including (but not limited to) the location, required functionality and asset management structure.

To demonstrate the concept of a DTOP, a specific example called DTOP-Cristallo has been presented in this paper. Readers can see a demonstration version, and get access to the code from GitHub (for details of how to do this, see the Data Availability Statement section). This example has been introduced to demonstrate the underlying programmatic structure and benefits for the different accessibility categories since DTOP-Cristallo is designed to be able be deployed in any of the three categories because of the generality and flexibility of Flask.

DTOP-Cristallo is designed for an example three-storey structure and is comprised of six separate tools that perform specific calculations that are useful for this type of system. This not only includes an analysis of the numerical model and experimental data, but also includes techniques for utilizing third-party software such as ABAQUS. DTOP-Cristallo is also available in a standalone format via the public GitHub repository.

In addition to the publicly available standalone version of DTOP-Cristallo, a server-based version has also been developed that makes modifications which are only possible due to the server-based structure. These upgrades primarily include distributed computing, remote access, and scheduling queues. Adding these upgrades greatly expands the usability and applicability of the DTOP for novel digital twin realizations.

Acknowledgment. The authors would like to acknowledge the support of EPSRC via grant number EP/R006768/1.

Supplementary Materials. To view supplementary material for this article, please visit http://dx.doi.org/10.1017/dce.2022.1.

Data Availability Statement. All the code and data presented in this paper can be accessed at the following GitHub repository: https://github.com/Digital-Twin-Operational-Platform/Cristallo. More practical implementation details can be found in the online documentation, which can be accessed via the GitHub page (see the link above).

Author Contributions. M.D.A., M.D.B., and D.J.W. created the initial framework of the DTOP using Python Flask. M.S.B. drafted this paper. All authors have contributed to the programming of the DTOP through individual tools and drafting the corresponding sections within this paper. D.J.W. provided project guidance for the DigiTwin project. All authors have read and approved the final manuscript.

Funding Statement. This work is funded by the EPSRC grant number EP/R006768/1.

Competing Interests. The authors declare that they have no conflict of interest.

References

Balatti D, **Khodaparast HH**, **Friswell MI**, **Manolesos M and Amoozgar M** (2021) The effect of folding wingtips on the worst-case gust loads of a simplified aircraft model. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* published online 22 April 2021, https://doi.org/10.1177/09544100211010915

Beregi S, **Barton DA**, **Rezgui D and Neild SA** (2021) Robustness of nonlinear parameter identification in the presence of process noise using control-based continuation. *Nonlinear Dynamics 104*, 885–900.

Chakraborty S and Adhikari S (2021) Machine learning based digital twin for dynamical systems with multiple time-scales. *Computers and Structures 243*, 106410.

Chatterjee T, **Adhikari S and Friswell MI** (2021) Multilevel decomposition framework for reliability assessment of assembled stochastic linear structural systems. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 7*(1), 04021003.

Copeland R (2008) *Essential SQLAlchemy*. California, USA: O'Reilly Media, Inc.

Cover TM (1999) *Elements of Information Theory*. New Jersey, USA: John Wiley & Sons.

Dal Borgo M, **Gardner P**, **Zhu Y**, **Wagg DJ**, **Au S-K and Elliott SJ** (2020) On the development of a digital twin for the active vibration control of a three-storey structure. In *Proceedings of the ISMA2020-USD2020*. Leuven, Belgium: KU Leuven.

DeAngelis M, **Behrendt M**, **Comerford L**, **Zhang Y and Beer M** (2020) Forward interval propagation through the discrete Fourier transform. In *The 9th International Workshop on Reliable Engineering Computing*. Risk and Uncertainty in Engineering Computations (virtual conference), pp. 27–38.

Eclipse Ditto (2021) Eclipse Ditto. Available at https://github.com/eclipse/ditto.

Elliott S, **Rohlfing J and Gardonio P** (2012) Multifunctional design of inertially-actuated velocity feedback controllers. *The Journal of the Acoustical Society of America 131*(2), 1150–1157.

Elliott S, **Serrand M and Gardonio P** (2001) Feedback stability limits for active isolation systems with reactive and inertial actuators. *Journal of Vibration and Acoustics 123*(2), 250–261.

Fuller A, **Fan Z**, **Day C and Barlow C** (2020) Digital twin: enabling technologies, challenges and open research. *IEEE Access 8*, 108952–108971.

Gardner P, **Dal Borgo M**, **Ruffini V**, **Hughes AJ**, **Zhu Y and Wagg DJ** (2020) Towards the development of an operational digital twin. *Vibration 3*(3), 235–265.

Gilchrist A (2016) *Industry 4.0: The Industrial Internet of Things*. New York, USA: Apress.

Gray A, **DeAngelis M**, **Ferson S and Patelli E** (2020) What's $z - x$, when $z = x + y$? Dependency tracking in interval arithmetic with bivariate sets. In *The 9th International Workshop on Reliable Engineering Computing*. Risk and Uncertainty in Engineering Computations (virtual conference), pp. 39–52.

Gray A, **Wimbush A**, **de Angelis M**, **Hristov P**, **Calleja D**, **Miralles-Dolz E and Rocchetta R** (2022) From inference to design: a comprehensive framework for uncertainty quantification in engineering with limited information. *Mechanical Systems and Signal Processing 165*, 108210.

Grinberg M (2018) *Flask Web Development: Developing Web Applications with Python*. California, USA: O'Reilly Media, Inc.

Jamia N, **Jalali H**, **Taghipour J**, **Friswell M and Haddad Khodaparast H** (2021) An equivalent model of a nonlinear bolted flange joint. *Mechanical Systems and Signal Processing 153*, 107507.

Jans-Singh M, **Leeming K**, **Choudhary R and Girolami M** (2020) Digital twin of an urban-integrated hydroponic farm. *Data-Centric Engineering 1*, E20. https://doi.org/10.1017/dce.2020.21

Jones D, **Snider C**, **Nassehi A**, **Yon J and Hicks B** (2020) Characterising the digital twin: a systematic literature review. *CIRP Journal of Manufacturing Science and Technology 29*, 36–52.

Kapteyn MG, **Pretorius JVR and Willcox KE** (2021) A probabilistic graphical model foundation for enabling predictive digital twins at scale. *Nat Comput Sci 1*, 337–347. https://doi.org/10.1038/s43588-021-00069-0

Liu H, **Chen W and Sudjianto A** (2005) Relative entropy based method for probabilistic sensitivity analysis in engineering design. *Journal of Mechanical Design 128*(2), 326–336.

Liu M, **Fang S**, **Dong H and Xu C** (2020) Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems 58*, 346–361.

Macedo T and Oliveira F (2011) *Redis Cookbook: Practical Techniques for Fast Data Manipulation*. California, USA: O'Reilly Media, Inc.

Minerva R, **Lee GM and Crespi N** (2020) Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE 108*(10), 1785–1824.

Momjian B (2001) *PostgreSQL: Introduction and Concepts*, Vol. *192*. New York: Addison-Wesley.

Myers J and Copeland R (2015) *Essential SQLAlchemy: Mapping Python to Databases*. California, USA: O'Reilly Media, Inc.

Niederer SA, **Sacks MS**, **Girolami M and Willcox K** (2021) Scaling digital twins from the artisanal to the industrial. *Nature Computational Science 1*(5), 313–320.

Nixon R (2014) *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5*. California, USA: O'Reilly Media, Inc.

Platenius-Mohr M, **Malakuti S**, **Grüner S**, **Schmitt J and Goldschmidt T** (2020) File-and API-based interoperability of digital twins by model transformation: an IIoT case study using asset administration shell. *Future Generation Computer Systems 113*, 94–105.

Preumont A (1997) *Vibration Control of Active Structures*, Vol. 2. New York, USA: Springer.

Rohlfing J, **Gardonio P and Elliott S** (2011) Base impedance of velocity feedback control units with proof-mass electrodynamic actuators. *Journal of Sound and Vibration 330*(20), 4661–4675.

Sacks R, **Brilakis I**, **Pikas E**, **Xie HS and Girolami M** (2020) Construction with digital twin information systems. *Data-Centric Engineering 1*, E14. https://doi.org/10.1017/dce.2020.16

Saltelli A (2002) Sensitivity analysis for importance assessment. *Risk Analysis 22*(3), 579–590.

Scheibmeir J and Malaiya Y (2019) An API development model for digital twins. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. Sofia, Bulgaria: IEEE, pp. 518–519.

Sieber J, **Gonzalez-Buelga A**, **Neild SA**, **Wagg DJ and Krauskopf B** (2008) Experimental continuation of periodic orbits through a fold. *Physical Review Letters 100*, 244101.

Wagg DJ, **Worden K**, **Barthorpe RJ and Gardner P** (2020) Digital twins: state-of-the-art and future directions for modeling and simulation in engineering dynamics applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering 6*(3), 030901.

Cite this article: Bonney MS, de Angelis M, Dal Borgo M, Andrade L, Beregi S, Jamia N and Wagg DJ (2022). Development of a digital twin operational platform using Python Flask. *Data-Centric Engineering*, 3: e1. doi:10.1017/dce.2022.1