EnDEVR: An Environment for Data Engineering in VR

Sarah M. Lehman [*] Temple University
Newton H. Campbell[†] NASA Goddard Space Flight Center
Simon A. Aytes [‡] University of Michigan - Dearborn
Mitchell Kirshner [§] University of Arizona
Anthony Arviola[¶] NASA Langley Research Center

ABSTRACT

Organizations within the public and private sectors are looking to improve their data science operations for research, development, and operational purposes. As this interest in data science grows, so too does interest in tools and programs that facilitating such operations. This paper presents EnDEVR, the Environment for Data Engineering in Virtual Reality, a user-extensible system that allows for the execution of custom data science functions. We describe the user-centered design process for the EnDEVR system, based on the needs of subject matter experts within the aerospace and aviation industries. Fundamental to this design is a method for users to integrate their existing analysis code (that may leverage third-party vendors) as usable objects in the virtual environment; this allows organizations to operate without the price tag and lifetime obligations associated with commercial data science toolkits, a concept known as "vendor lock-in". We then present our system design with iterative implementations, and conclude with lessons learned. The presented work may provide insight into future user-centered design processes for VR applications, particularly for digital transformation stakeholders to circumvent vendor lock-in.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual Reality; Human-centered computing—Visualization—Visualization systems and tools; Human-centered computing—Visualization—Visualization application domains—Visual analytics

1 INTRODUCTION

Organizations in every sector are going through a digital transformation, in which they deliberately integrate new digital technologies into existing processes to improve how they operate and deliver value to customers [1, 34]. Virtual reality (VR) is one such technology that has received increasing attention in a variety of domains, such as education [27, 43], healthcare [13, 41], manufacturing [7, 31], tourism [28, 60], and the space sector [29]. VR demonstrates promise as a tool for data analysis and exploration thanks to its ability to display arbitrary, interactable 3D visualizations [18, 39]. This makes VR an exciting candidate for organizations undergoing digital transformation to elevate their data science practices.

Data science-focused digital transformation is nearly always paired with third-party services and libraries for data engineering and state-of-the-art artificial intelligence [48]. Libraries such as Scikit-Learn for Python [49] and Google's ML Kit Application Programming Interface (API) [9] offer prepackaged functionality for users to incorporate directly into their code. Tools such as IBM Cloud Paks [25], Splunk [53], and Tableau [55] provide graphical, "pipeline-based" interfaces for data scientists to explore and interact with their data sets. However, organizations who seek to use these tools often risk coupling their own software implementations tightly to a vendor's API or service. Consequently, if these organizations do not make incredibly nuanced purchasing decisions, they can incur high operating costs for these toolkits while simultaneously limiting their data science capacity to that of the vendor. These effects can be more impactful in organizations that do not have a trained data science staff on-hand to sufficiently augment vendor capabilities. Even open-source frameworks are often optimized for their commercial cloud offerings [6]. In many cases, this can tie a project's framework to a vendor for life, a concept called "vendor lock-in". Vendor lock-in provides two expensive options to an organization: either accepting the high cost and low level of modifiability of these commercial tools, or developing custom tools themselves.

Organizations wishing to improve their data science infrastructure in a way that leverages new API and services, but avoids vendor lock-in, face the following challenges:

- **(C1)** Because data science practices vary between industries, potential users must be an integral part of the data infrastructure engineering process to ensure the acceptability of any subsequent system design.
- **(C2)** Because data science teams may use any number of tools or algorithmic implementations to achieve their goals, the infrastructure needs to be user-extensible (e.g. support an open-ended collection of analysis operations, native code, third-party libraries, and other processing capabilities).
- **(C3)** Because data science operations can be highly complex, any new data science tools introduced to the infrastructure must provide enough novelty and utility to justify the learning curve and labor investment involved in adoption. Otherwise, a data scientist will likely stay with their current tool suite.

With these challenges in mind, we present the EnDEVR system, an Environment for Data Engineering in Virtual Reality. EnDEVR is an applied mathematics and data science ecosystem that allows users to construct and investigate customizable data analyses from a VR environment. This system addresses vendor lock-in issues among data science toolkits while promoting VR as a viable data science platform in the aerospace and aviation industries. The principal characteristic of the system is user extensibility, in which users can submit custom or prototype algorithm implementations, and employ them in VR data processing pipelines. A dedicated computing server then processes the pipelines, generating VR visualizations of the results for users to inspect in 3D. Users can also export these pipelines as computer code to include in their mission workflows. Because users submit them, these pipelines can leverage any number of accessible third-party libraries, systems, or API, reducing operational dependency on any individual vendor and avoiding vendor lock-in.

This paper makes the following contributions:

- **A user study on the data analytics practices and VR acceptance criteria among scientists and engineers in a prominent aerospace organization.** This addresses C1 by providing guiding principles and design points to ensure acceptibility of the final system.

[*]e-mail: smlehman@temple.edu
[†]e-mail: newton.h.campbell@nasa.gov
[‡]e-mail: saytes@umich.edu
[§]e-mail: mkirshner@email.arizona.edu
[¶]e-mail: tony.arviola@nasa.gov

- **A proposed system for aerospace professionals to conduct data science operations in VR.** This addresses C2 by incorporating specific design points for handling of open-ended algorithmic implementations, data sources, and high level workflows.
- **A presentation of lessons learned while implementing the system within the context of commercial and government infrastructure.** This addresses C3 by leveraging practical experience to identify barriers to adoption within our industry.

The rest of the paper is organized as follows. Section 2 discusses the study background and related work; Section 3 describes our user study; Sections 4 and 5 describe our system design and implementation demonstration respectively, while Section 6 discusses our lessons learned and Section 7 concludes.

2 Related Work

The ability to interact with data in a 3D environment can allow users to understand complex and disparate data sets more readily than 2D methods. NASA Jet Propulsion Laboratory (JPL) demonstrated the utility of commercial-off-the-shelf (COTS) VR hardware to generate a 3D environment from 2D images of Mars taken by the Curiosity rover, allowing for more significant knowledge discovery [11]. Other work has identified increased feelings of satisfaction and success, fewer errors, and comparable mental effort among testers when using VR over 2D graphs [35]. Research within the aerospace industry has verified similar results in VR and augmented reality (AR) for data visualization, highlighting the cognitive bottleneck between data and discovery in a 2D environment [3, 18, 22, 36, 37]. Adequate visualization of high-dimensional feature spaces is a critical challenge addressed by operating in 3D spaces [14]. However, the need for algorithms that generating 3D visualizations of arbitrary data has been identified as an open research topic, with several proposed frameworks [39, 50].

Digital transformation and machine learning are tightly coupled concepts in the aerospace and aviation communities. These communities have verified the utility of VR in design and system verification and validation [30, 56, 59]. The aerospace community is currently focused on the problem of communicating with and maintaining assets in remote, potentially hazardous conditions [32,45,46]. Space weather has also become a target for new machine learning research [4,12]. In the aviation sector, the integration of physics-informed machine learning and high-quality modeling for assured, autonomous decision-making is a significant research area [20, 21, 38]. The ability to use VR for design and verification, paired with the growth of machine learning applications to aerospace and engineering, makes the ability to command diverse machine learning technologies from VR a logical next step for the research community.

Enterprise services like Tableau [55] or IBM Cloud Paks [25] advertise robust, high-compute performance. However, they offer limited control or modifiability of the internal algorithmic logic, with no current support for advanced user interfaces such as VR. While some services like Splunk have begun to offer support for 3D visualizations in VR [53, 54], users are limited in the operations they may perform based to what is offered by the vendor API provides. Open-source platforms such as Orange [40] allow for algorithm customization, but require both a significant understanding of the framework's codebase and recompiling of the entire codebase to introduce a new algorithm.

2.1 Enabling Technologies

A VR game engine is built with low-level control logic in-place, allowing simplified actions to create, place, and animate objects within a 3D space, and reducing the need to develop this type of code from scratch. Given their support and communities, the top two candidate engines for developing the EnDEVR system were Unity [24] and Unreal Engine 4 [15]. Various comparisons have been made between the two [10, 58]. Both have benefits and drawbacks, but there is currently no community consensus on the general utility of either.

Both game engines permit limited use of common libraries for their native programming languages (C# for Unity, C++ for Unreal Engine). To overcome this limitation, real-time communication with other processes and systems with expanded capability is necessary. For the EnDEVR communication framework, we explored three enabling technologies. The integration of native sockets with a visual pipelining framework has previously been explored [26], but requires significant socket management for implementation. gRPC is a high-performance framework for Remote Procedure Calling (RPC) that communicates using HTTP/2 [8, 33]. The framework supports high-performance streaming between clients and servers, but does not integrate well with VR game engines. gRPC works directly with protocol buffers, a language and platform-neutral data interchange mechanism for serializing structured data [19]. Finally, the Websocket protocol operates over HTML5 and supports chat-like communication between client and server [16]. It has demonstrated a capability for fast, real-time data communication between systems [42, 44]. Both Unity and Unreal Engine have third-party libraries that permit Websocket communication.

3 Stakeholder Requirements Gathering

User-Centered Design (UCD) is characterized by the early adoption of user needs as guiding principles in the design process [47]. This initial focus on end-users enables a team to make informed design choices that meet the organization's needs at large. To identify user needs for the EnDEVR system, we began development with a series of stakeholder interviews. The results of these interviews guided the requirements for design, implementation, and testing. By employing UCD from the beginning, we avoided common design pitfalls that arise when creating and deploying an application at-scale. In the following section, we discuss our user study methodology, results evaluation, and key takeaways.

3.1 Interview Methodology

The stakeholders who participated in our interview process consisted of industry subject-matter experts, and potential end-users for our system. We manually selected participants from within our organization based on availability, interest, and position in the organizational hierarchy. The final group of stakeholders (n=20, five female) represented seven business centers within our organization and had a combined total of more than 200 years of industry experience.

Each stakeholder was provided with a description of the proposed system concept ahead of time. The interview session began with a 15-minute presentation and system demonstration, consisting of a high-level overview of the EnDEVR concept,its relation to the organization, and potential applications. We followed with a semi-structured interview with a free-form discussion. We asked each stakeholder about their current data analysis tools and procedures, any challenges they face with those tools and procedures, and their prior experience with VR systems. Each interview session lasted one hour and was recorded with consent.

We selected questions with an understanding that adopting our system would require teams or business centers to adopt both VR and a new machine learning framework; thus, questions were crafted to improve understanding of current data analysis processes and experiences in VR. In additions, during the free-form discussion, nearly every stakeholder introduced ideas about potential organizational use cases that would require specific features to be added to the original concept of the system.

	Scope	Time to Learn	Difficulty	Scalability	Appeal	Freq. of Use	Originality	Alignment
Total	16.17	20.5	19.83	10.83	10.83	15.92	21	9.5
Average	2.31	2.93	2.83	1.55	1.55	2.27	3	1.36
Std. Dev.	1.06	0.89	0.9	0.81	0.81	1.02	1.08	0.48

Table 1: Stakeholder feedback values with associated scores.

DRIVER	DESCRIPTION
User Extensibility	EnDEVR will provide the ability for end users to submit custom algorithms for use in workflow construction in virtual reality.
Open Source	EnDEVR will provide the ability for future developers to expand system functionality.
Usability	EnDEVR will provide intuitive, interactive, and accessible user interfaces within the virtual reality environment.
Functionality	EnDEVR will include built-in functions to provide out-of-the-box processing and use-case demonstrations on top of a steel-thread implementation.
Security	EnDEVR will meet the requirements of internal organizational software tools.
Performance	EnDEVR will operate within reasonable time and computing constraints.
Reliability	EnDEVR will be able to reproduce experiences and visualizations between user sessions.
Portability	EnDEVR will be easily distributed and installed on supported devices and operating systems.

Table 2: Architectural drivers for the EnDEVR system design, elicited from stakeholder interviews.

3.2 Value Scoring

The purpose of the stakeholder interviews was to identify desired system characteristics, functionality, and use cases to incorporate into the final system and ensure acceptability and adoption among end-users. Following each interview, these points were extracted from the stakeholder responses and individually evaluated by each interviewer according to the following values: scope of the work to be done with the system, the time to learn the system, difficulty of using the system, scalability, general appeal, anticipated frequency of use, originality of the system concept, and alignment with team goals. Each interviewer scored the stakeholder's responses on a 5-point Likert scale for each of these values, where 1 was the best score and 5 the worst. Scores for each category were averaged across interviewers, while the sum of scores across categories represented the final score for each stakeholder (summarized in Table 1).

The lowest-scoring stakeholders reflected those suggestions and priorities aligned most closely with EnDEVR's scope and concept, and so were prioritized in the subsequent system design. The data from the initial interview analysis showed a clear trend among the top three lowest-scoring stakeholders: they each received perfect scores for Scalability, Appeal, Accessibility, and Alignment. Results imply that the initially proposed concept for EnDEVR was aligned with key stakeholders in our organization. Conversely, Time to Learn and Difficulty score were among the highest, which is reasonable given the small size of the EnDEVR development team. The final scores were summarized and presented to project leadership. The final list of stakeholder needs, and their corresponding scores, were used to develop the system architectural drivers described in Table 2, and eventually, the initial set of system requirements.

3.3 Development of System Design Guidelines

The information gathered during the stakeholder interviews revealed a wide variety of tools and practices used within our organization. Stakeholders reported their teams engaging in data science operations such as reinforcement learning, using neural networks to classify text and image inputs, natural language processing, variable correlation, and autoencoding. They also reported using tools such as Python and Matlab data science libraries, dedicated machine learning SDKs such as TensorFlow and OpenCV, cloud-based services such as Google Cloud Platform and Apache Spark, Jupyter notebooks, and Watson Explorer. They also reported managing data sources in various formats, such as raw text files (e.g., plain text, XML, CSV), spreadsheet documents, Hierarchical Data Format files, PDFs, and images. All stakeholders were all receptive to adopting VR as a tool for their work, provided appropriate precautions were taken to address motion-sickness factors, and to make the system as accessible as possible. Additionally, they all expressed interest in the potential uses of EnDEVR in their respective fields.

As a result of the stakeholder interviews, we identified three recurring, high-level needs that EnDEVR would need to address:

- **The ability to perform free-form data exploration**: This need was mentioned many times by many different stakeholders. Scientists in our organization routinely have to perform open-ended knowledge discovery operations on vast, disparate data sets, so tools to improve that process would be welcome.
- **The ability to process, visualize a wide range of high-dimensionality data**: In our organization, data sets take many different forms and can represent a spectrum of variables and dimensions. When considering VR as a platform for visualizing data sets, one stakeholder remarked, "What is intriguing to me is the other quantities such as training an algorithm that impact [our understanding of] vehicle dynamics, and being able to display the weights of a neural network at the same time as displaying the altitude and movement of an aircraft."
- **A mechanism to securely author, reuse, and share algorithms and data sets between users**: Another recurring theme among stakeholders was reducing redundancy by sharing and reusing both data sets and algorithmic implementations between team members and throughout the organization. Stakeholders additionally reported being dissatisfied with the current data management guidelines within the organization, the cleanliness of the data they received, and the format of that data.

Using these insights in combination with the manual value scores, we developed architectural drivers for the EnDEVR system (summarized in Table 2). These drivers guided system design and provided heuristics to verify the utility of the system in our organization.

4 System Design

The EnDEVR system design centers around the idea of a *workflow*, which is a structured set of computable tasks, called *widgets*. Each widget represents a custom algorithmic implementation to be executed, and can be comprised of any combination of native code and calls to third-party libraries, APIs, or external services. A connection between widgets indicate that a given widget accepts the output of a

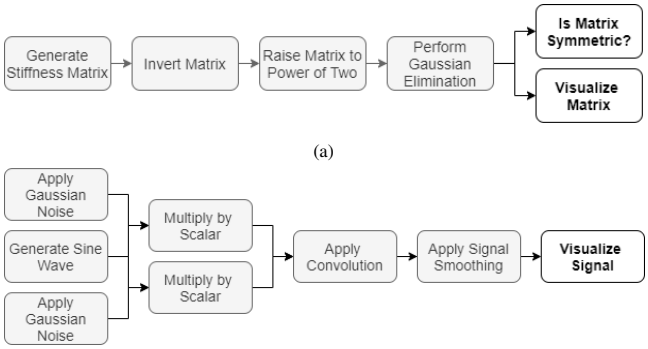


Figure 1: Example data science workflows, where composite nodes (grey) perform algorithmic operations and leaf nodes (white) return results or visualizations.

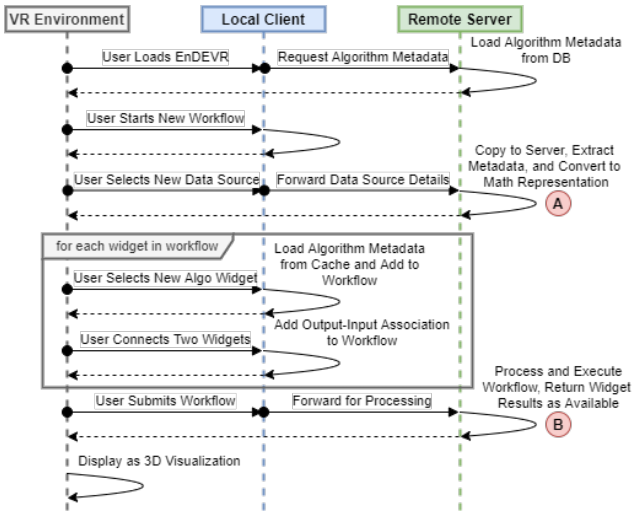


Figure 2: Sequence diagram for creating and executing a workflow using the EnDEVR system.

prior widget as an input in its own computation. Users in virtual reality construct workflows by selecting widgets from a list of available options and connecting the widgets into a pipeline. Two examples of workflows that could be built with the EnDEVR system are shown in Fig. 1. The user can then command the system to forward the workflow to a remote server, which executes the selected operations and returns corresponding results for 3D visual assessment. This process is reflected in Fig. 2.

4.1 System Architecture

The EnDEVR system design is an N-tier system architecture (N=4) [23], as reflected in Fig. 3. The tiers are organized as follows:

- **Presentation Tier:** comprised of the VR environment and all related user-interface logic. Manages the user's ability to move and interact with elements in the environment, including selecting and configuring widgets; displaying workflow status and error messages; and displaying and interacting with workflow output visualizations.
- **Control Tier:** coordinates the presentation tier, executes commands or operations that the VR game engines are incapable of, and sends workflows to the processing tier. Serves as a proxy between the Presentation and Processing tiers.

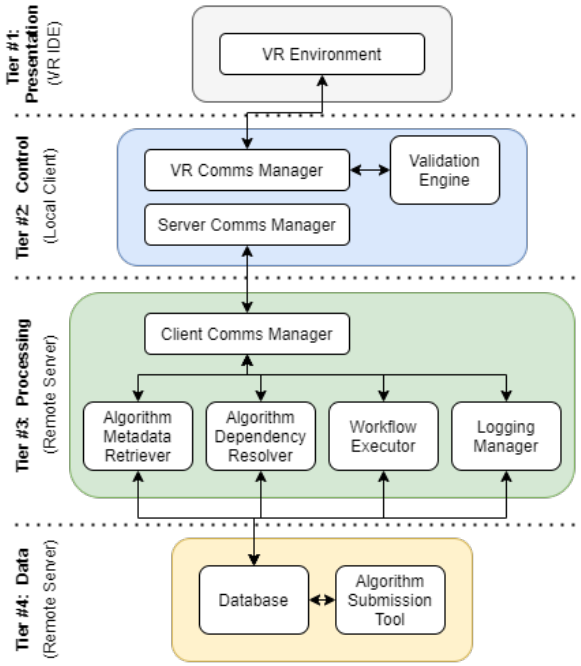


Figure 3: EnDEVR architectural model

- **Processing Tier:** receives and executes commands from the control tier. Principally responsible for downloading code for workflows from the data layer, arranging and executing this code according to the workflow structure, and streaming results back to the control tier. More information about this tier is presented in Sections 4.2 and 4.3 respectively.
- **Data Tier:** maintains algorithms and corresponding code available for use in workflow construction, as well as general author, user, system logging, and role-based access control information. Provides a standalone GUI to allow users to submit new algorithms. Algorithm submission is covered in greater detail in Section 4.4.

This architecture maps to the architectural drivers of Table 2 as follows. The VR environment at the Presentation tier grants the system usability and functionality. After reviewing alternatives, we selected candidate technologies at this tier that would allow for out-of-the-box, intuitive usage. By placing a Control tier between a VR environment and backend server, the architecture addresses the reliability driver by ensuring the system still has some local capability if the connection with a backend server is lost. The design of the middle two tiers also accounted for connections being lost and recovered. The server tier permits both system portability and performance by minimizing dependencies for local users and pushing the bulk of responsibilities to the server. The Data tier provides user extensibility and security by permitting the user community to submit algorithms and share them with authorized users. Finally, we documented the process for source control, documentation, and logging/debugging for this modular architecture to promote adoption and contribution to the open-source community.

4.2 Data Source Handling and Representation

When the user starts the EnDEVR application, the system creates a digital thread across all four architecture tiers. Fig. 2, mark (A)

Algorithm 1: Data Source Loading and Processing

```
   Input: Data location L, Location type T
   Output: Math representation r
   /* acquire data source                                   */
1  if T is local filepath then
2  |  file ← uploadToServer(L)
3  end
4  else if T is remote URL then
5  |  file ← downloadToServer(L)
6  end
   /* process data source                                   */
7  storeInTempDirectory(file, session.ID)
8  metadata m ← extractDataCharacteristics(file)
9  mathRep r ← convertToMathRep(file, m)
10 return r
```

depicts this procedure. After selecting a *data widget* within the EnDEVR Pipeline Builder, the user specifies a local or remote path as a data source and passes the path information to the Control tier. If the path is local, the Control tier uploads the data to the Processing tier. If not, the Processing tier will be sent the remote path URL and will download the data to temporary storage. When the user submits a workflow through the Presentation tier, the system embeds references to the data in temporary storage in the instructions for processing. This digital thread allows the system to avoid using the user's local machine processing power to upload data directly into the VR environment.

When the user uploads data to the Processing tier, the Control tier receives a reply as a stream of mathematical representations of the data, which the VR algorithmically visualizes. The standard EnDEVR representations include: matrices, sets, sequences, strings, graphs, numbers, functions, probability distributions, and tuples. A data widget can only produce the first four of these representations. The rest are outputs of custom algorithm widgets. These structures were selected based on a review of key mathematical objects and notations used in automata theory [52] and abstract algebra.

The Presentation, Control, and Processing tiers have a common understanding of these representations. For example, the Presentation tier does not handle raw data. Instead, it only ever visualizes or allows the user to explore one of the representations derived from the data (with visual mapping to the original data specification). The Control tier performs validation checks on a workflow based on these representations, and the Processing tier translates between the inputs and outputs of widgets and these representations when processing a workflow. The messages to each tier, containing these representations, include metadata that are the byproduct of rudimentary analyses (e.g. mean, variance, standard deviation).

4.3 Workflow Processing

The central use case of EnDEVR is for a user at the Presentation tier to construct a workflow and submit it for processing. At the Presentation tier, the workflow exists as simple mappings of inputs to outputs. The workflow is logically validated at the Control tier and forwarded to the Processing tier, which runs the referenced algorithm implementations in order. Once the user submits a valid workflow for processing, the server is responsible for parsing that workflow into a set of executable steps with the appropriate data flow between steps (shown in Fig. 2, mark Ⓑ).

In processing, we formulate a workflow as a directed, acyclic graph (DAG), $G = (V, E)$, where V is a set of identifiers for algorithm implementations and E is a set of V's edges. Without loss of generality, for a workflow G, if $(u, v) \in E$, then the output of the algorithm referenced by u is one of the inputs of the algorithm referenced by v. G must also contain one or more sources and sinks

Algorithm 2: Workflow Processing and Execution

```
   Input: Workflow object W
   Output: Set of math representations (returned async)
   /* convert workflow to directed acyclic graph            */
1  for widget w in W do
2  |  add node(w) to G
3  |  for existing node(v) in G do
4  |  |  if W contains out-in dependency(v, w) then
5  |  |  |  add directed edge(v, w) to G
6  |  |  end
7  |  end
8  end
   /* traverse graph by level and execute                   */
9  nodeGroup ← getSourceNodes(G)
10 nodeOutputs ← { }
11 while nodeGroup is not empty do
12 |  for node n in nodeGroup do
13 |  |  node[] p ← getParentNodes(n)
14 |  |  output[] in ← nodeOutputs.getForNodes(p)
15 |  |  output out ← executeStoredCode(n, in)
16 |  |  nodeOutputs.add(n, out)
17 |  |  returnAsync(out.toMathRep())
18 |  end
19 |  nodeGroup ← getImmediateChildNodes(nodeGroup)
20 end
```

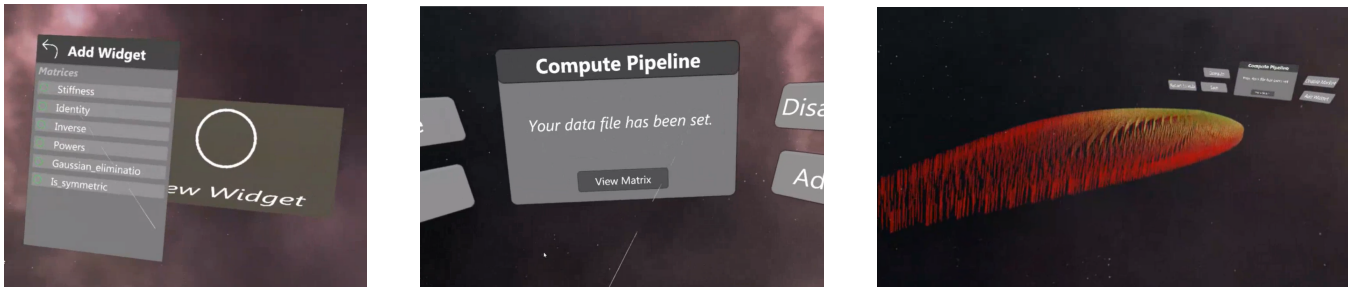
for the workflow to be processed by the system. Each widget in the workflow is represented by a vertex $v \in V$ with its associated executable code.

Algorithm 2 describes the steps for workflow processing at the Processing tier in two main phases. In the first phase, we convert the workflow mapping of inputs and outputs into a logical DAG (lines 1-8). Each widget in the workflow is added to the graph with directed edges indicating that a node's output is used as an input for another. The Processing tier queries the executable code and dependencies from the Data tier. In the second phase, the system traverses the graph by layer and executes the code for each node in the layer in parallel (lines 9-20). As each widget completes computation, results are converted into a system math representations and sent back to the Control tier immediately, passing them to the Presentation tier. These same results are serialized and stored as a temporary file on the Processing tier server using a language-based serialization library (e.g., Pickle [57]). Subsequent widgets that need these results as input deserialize these files and read their contents. This process continues until all sinks in the tree have been processed.

4.4 Algorithm Handling and Storage

Users specify algorithms for the EnDEVR system as function-level scripts, stored in the Data tier as binary snapshots. The Data tier stores additional metadata for each algorithm, such as the function name, a user-friendly display name for reference in the VR environment, a description, a functional category, and the corresponding author. Library dependencies and information about expected inputs and outputs are also stored. The Data tier supports custom code written in Python, R, and other data analysis scripting languages and calls to third-party services and APIs.

Fig. 5 shows an example of an acceptable algorithm for the Data tier in Python. It contains source code and a Docstring describing the function, parameters, output value, and any errors raised. Any libraries that the function depends on are listed as function-level `import` statements. The function, as a file, is uploaded to the Data tier via the Algorithm Submission Portal, which walks the user through the steps of providing the metadata mentioned above. The portal automatically extracts some information directly from the



(a) Selecting Widgets

(b) Submitting the Workflow

(c) Visualizing Results

Figure 4: Screenshots of the EnDEVR steel thread.

```python
def sine(angleDeg):
    """
    Calculates the sine value for a given angle

    :param angleDeg: the angle in degrees

    :return the sine value
    """

    import math
    result = math.sin(angleDeg)
    return result
```

Figure 5: Sample algorithm formatted for submission to EnDEVR.

function, such as the function description and list of dependencies. Other information has to be manually provided by the user. Once the user provides all of the requisite metadata, the function file is stored directly as a binary snapshot, and the algorithm is ready for use in workflow construction.

Organizing and storing algorithms in this way serves several important functions. It supports the User Extensibility and Open Source architectural drivers by allowing users to submit their own algorithmic implementations that exactly meet their own unique goals; it supports the Security architectural driver by identifying algorithmic authorship for access control; and it lessens user reliance on individual data science libraries or services by allowing users to submit multiple implementations for the same algorithm if project needs or vendor functional offerings change.

5 System Implementation

To refine acceptance criteria for a full EnDEVR system, over four months, we developed a "steel-thread" implementation [2], containing the basic functionality of the system. The lessons learned from demonstrating this system to stakeholders across our organization were incorporated into the design and development of the full EnDEVR system.

5.1 Steel Thread

The steel-thread contained the main functionality for each tier of EnDEVR, using native socket communications. The Presentation tier was implemented as an application developed in Unity [24], designed for use with the Oculus Rift-S. The Control and Processing tiers were implemented as C# client and server applications, respectively. Our organization provided a small CentOS virtual machine (VM) on an enterprise business server to host the Processing tier. We also were provisioned 2TB of space from the organization's existing managed MySQL database server.

The steel-thread Presentation, Control, and Processing tiers communicate using C# TCP/IP sockets. The Control tier application communicates with the Processing tier VM through an SSL (Secure Sockets Layer) tunnel connection over the organization's internal network. All communication among the tiers was in the form of JSON (JavaScript Object Notation) messages. When a user initiates the VR application, the Control tier is started and connects to the Processing tier server. The algorithm submission tool for the Data tier was implemented as a simple command-line C# tool.

Fig. 4 shows some screenshots of the steel-thread during runtime. Once in the VR environment, the user can upload a new dataset or resume work on an existing dataset. A request is then made for available widgets to the Data tier. The user selects, moves, and connects widgets using hand gestures (Fig. 4a). When the user submits a workflow (Fig. 4b), it is sent to the Control tier, parsed into a JSON object, and forwarded to the Processing tier. The Processing tier then deserializes this object, requests the scripts from the Data tier database, and assembles the scripts according to the current workflow. The system calls Python or a custom executable based on the assembled scripts. Results are then returned to the Control tier, read through the Unity API and visualized in the VR environment (Fig. 4c).

5.2 Full System Implementation

By evaluating the steel-thread with stakeholders, we identified several key issues to address in implementing a full system. First, due to the nature of our organization, Unity's licensing model was not cost-effective for adopting new developers across the organization to our team. Unreal Engine provides a more cost-effective licensing model for our organization to grow the EnDEVR system and team. In addition, Unreal Engine's asset marketplace tends to offer visualizations that are of higher visual quality [10]. A stakeholder comparison verified these results, and we adapted the mid-level design of the system to Unreal Engine.

Second, we reimplemented the Control and Processing tier applications in Java, for its platform-independence and concurrency features necessary for growing the system's user community within our organization. In addition, the steel-thread relied on a set of custom-defined JSON messages, which required redefining these messages for the application at every tier. For communication between tiers, the system uses Protobuf, WebSockets, and gRPC. Messages are serialized as protobuf messages, defined once and understood by all tiers. The VR environment communicates with the Control tier client on the same local machine, passing Protobuf messages via interprocess communication (IPC) over WebSockets. Communication between the Control tier and the Processing server application occurs using gRPC. A full system diagram reflecting these changes is illustrated in Fig. 6.

Finally, stakeholders indicated that we needed a straightforward mechanism for submitting algorithms to the Data tier. For this, we developed a set of database-enabled Jupyter notebooks for data analysts to use when submitting existing scripts or prototyping new analyses. The algorithm submission notebook presents a traditional form-based GUI which walks authors through the process of loading

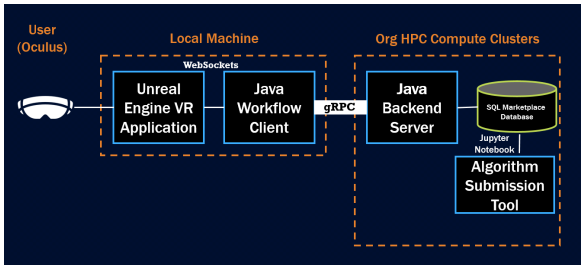


Figure 6: EnDEVR full-system implementation diagram.

Representation	Visualization	
Sets	Word/Symbol Cloud	
Matrices	Parallel Coordinates Plot	
Sequences	N-D Scatter Plot	
Tuples	Scatter Plot	Heat Map
Functions	Equation Text	
Graphs	Graph Drawing	
Numbers	Histogram	Complex Number
Distributions	Density Plot	
Strings	Word Cloud	

Table 3: Mapped visualizations for EnDEVR math representations.

an algorithm for submission, extracting the key metadata, performing some preliminary validation, and finally submitting it to the database.

5.3 Evaluation

Objectively evaluating EnDEVR is admittedly difficult. There is no benchmark test suite where we can compare our results against comparable systems. Therefore, we solicited the stakeholders within our organization analysis use cases that would be ideal for the environment. We chose three use cases, focused on intelligent aircraft, space weather, and independent verification & validation (IV&V) of systems, respectively. We implemented the intelligent aircraft use case in the steel-thread and all three in the full system implementation.

The steel-thread was evaluated on an experimental eVTOL (electric vertical takeoff and landing) vehicle model known as Lift-Plus-Cruise [51]. eVTOL has become the focus of the latest research in urban air mobility and intelligent flight. The stakeholder's group was researching the use of kinetic and potential energy estimation for on-board assessment of the vehicle. In coordination with the subject matter experts, we developed and submitted the algorithms necessary for running a MATLAB autoencoded simulation of the Lift-Plus-Cruise in flight. For this use case, we developed scripts for three widgets: the first to allow a user to parameterize the mass of various parts of the vehicle, the second for running the MATLAB simulation with specific trajectories, and a third that takes in the simulation outputs and computes the kinetic and potential energy of the rigid-body system during flight. We verified the results in the VR environment against those on a standalone laptop for correctness. While this was the first successful instance of commanding and analyzing an external simulator from the environment, the 3D rendering of the result data was unintelligible. This led to developing algorithms for plotting multiple time series as shown in Fig. 7 and the system mapping of math representations to corresponding visualizations shown in Table 3.

For the second use case, we demonstrated the analysis of issue reports submitted to the IV&V group. These issue reports describe problems in project development, covering topics from inconsistent design to missing code. Current practice for mining these reports centers on extensive manual investigation and filtering efforts, resulting in significant duplication of effort between analysts. We

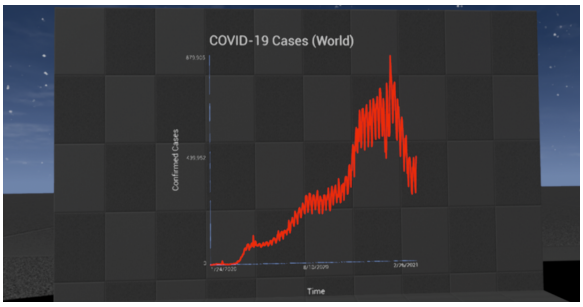
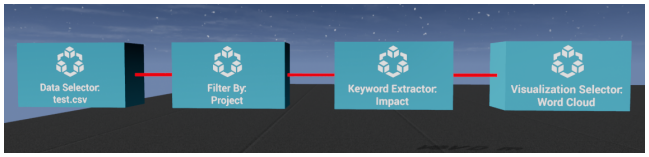
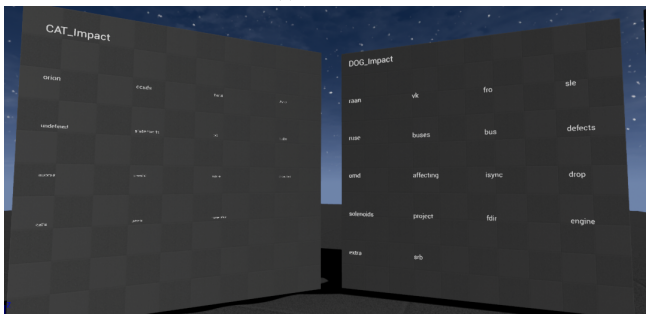


Figure 7: Time series plot in EnDEVR full system implementation.



(a) Workflow



(b) Visualization

Figure 8: IV&V use case screenshots.

implemented new EnDEVR widgets to generate word clouds based on descriptions of issues logged for a given project (example workflow shown in Fig. 8a). Stakeholders indicated that this kind of visualization in VR would be significant to their mission of identifying project-level trends. For this use case, we implemented algorithms for three new widgets, and published them for use via the Algorithm Submission Toolkit (the results of which constituted our first demonstration of the system for text-based data and guided subsequent visualizations for natural language processing):

- **Field Filter**: Accepts a 2D matrix of data, and filters based on a selected condition. For our use case, we subdivided the data set by unique values in a particular field (e.g. project name).
- **Keyword Extractor**: Applies document summarization and N-gram keyword extraction to string data. For this use case, it returns the top K keywords and their associated weights for each data subset.
- **Word Cloud Visual**: Generates a word cloud for a given set of keywords and their associated weights. We generated a "cloud of clouds" for this use case where each data subset received its own cloud (2D examples shown in Fig. 8b). Because label placement in 3D space is a complex problem space [5], we are leaving more robust cloud display logic for future work.

For the third use case, our Heliophysics research group identified magnetic disturbance data from US NOAA [17] that would be suitable for submitting machine learning algorithms to the Algorithm Submission Toolkit. As NOAA had previously hosted a

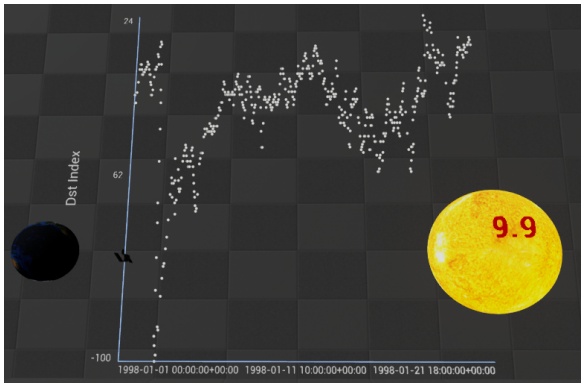


Figure 9: Magnet use case visualization screenshot.

competition for this data with publicly available code, we used this to demonstrate the ability of the system to incorporate open-source code. We downloaded open-source analysis code for predicting disturbance storm-time indices (DST) using solar wind data, and used it to develop 15 new algorithms from this open-source codebase, many of which have strong potential for reuse outside of the space weather application. This conversion process from open-source code to algorithms suitable for the EnDEVR system was completed by an individual Computer Science PhD student over approximately 3 work days (24 hours). They noted difficulty interpreting and removing dependencies within the open-source code. Much of their time was spent writing descriptions of each widget and figuring out how to make changes to remove those interdependencies.

6 Discussion

In the following section, we will discuss our lessons learned throughout the process of implementing EnDEVR, as well as a selection of open problems and future work.

6.1 Lessons Learned

Our takeaways from evaluation of the EnDEVR system fit into several categories that will guide future developments: system innovation, general system robustness and error-handling, client/server environment configuration, code structure and organization, and documentation.

System Innovation: Numerous system innovations became apparent as EnDEVR approached completion. The 3D widgets used in workflow building allow for practical data engineering in VR, allowing users to incorporate diverse APIs in separate implementations, thus avoiding vendor lock-in. The Processing tier specifically demonstrated its ability to handle numerous types of data and algorithms to effectively compute pipelines developed for data science and engineering operations. Behind the scenes, the WebSocket and gRPC connections between tiers provide an effective communication vehicle for seamless real-time processing. The use of WebSockets with the VR front-end will be a key system feature going forward.

Robustness and Error-handling: Several lessons were learned for general system robustness and error-handling that should be applied to future development. When transferring CSV files between tiers using gRPC protobuf serialization, empty values cause errors. Therefore, the data being sent must be fully populated with values or use a special value denoting missing data. This may cause more data to be sent than necessary and can effect real-time processing. On the algorithm submission side, successful validation of submitted Python scripts required strict formatting constraints which were communicated to the algorithm author prior to submission.

Environment Configuration: As part of the artifacts created during system development, our team documented a set of guidelines for environment configuration and hosting. This included a list of required software installations, required versions and dependencies, and environment variables to ensure correct inter-tier communication using WebSockets, gRPC, and C++/Java executables. As a general rule, when setting up the EnDEVR environment, adherence to organization-level expectations for security, availability, accessibility must be considered during all phases of system development and use. This is especially true using internal hosting platforms, which can help reduce computational load bottlenecks.

Code Structure, Organization, and Documentation: We developed consistent development processes for maintaining uncluttered code structure and organization. Eclipse, Visual Studio, and Unreal Engine software facilitated handling of the large number of files used to create EnDEVR. Maintaining consistent code structure and organization across the multi-tier system was important, as (at the time of this writing) Unreal Engine 4.26.2's capacity for source control with Git is still fairly immature.

6.2 Open Problems and Future Work

EnDEVR is an on-going project. There are a number of open problems that this first phase introduced and that we plan to address in future work. First, we will explore additional methods for mapping arbitrary data sources to the systems math representations. In large organizations, institution-level conventions regarding data structure and storage practices are generally lacking, leading to data storage form factors ranging from CSVs to raw text documents to images of spreadsheets. As such, algorithmically understanding the nature of these data sources and developing appropriate representations remains an open problem.

Second, we will explore methods for more stringent validation and verification of submitted algorithms and workflows. Algorithms must be syntactically and semantically verified, while workflows must be examined to ensure compatibility of output-to-input connections between widgets. Additionally, graceful error handling as well as suggested improvements for workflow organization and unexplored analytical operations still need to be investigated.

Finally, we will explore access control and security measures to mediate sharing of algorithms and data sources within our organization. Authentication and authorization are significant concerns for all organizations in both public and private sectors, and are made even more important when large-scale, mission-critical, and export-controlled data stores are involved. Further, any security measures that are implemented should not only meet end-user and project-level expectations for privacy and access control, but must also conform to any regulations set forth by the parent organization.

7 Conclusions

In this paper, we presented Environment for Data Engineering in Virtual Reality (EnDEVR), an applied mathematics and data science ecosystem that allows users to command and investigate customizable data analyses from a virtual reality environment. A key feature of the Environment for Data Engineering in Virtual Reality (EnDEVR) ecosystem is that it is not dependent on any one proprietary data science platform, allowing users to extend its data analysis capabilities without restriction. We presented a user study of data science practices among expert stakeholders in the aviation and aerospace industry, and used that feedback to drive the implementation of EnDEVR, first as a "steel thread", then as a full system implementation with three aerospace and aviation use cases. Finally, we discussed our lessons learned and a selection of open problems. In future work, we will explore additional human factors to improve the user experience and physical comfort when using EnDEVR.

Acknowledgments

The authors wish to thank A, B, and C. This work was supported in part by a grant from XYZ.

REFERENCES

[1] U. G. S. Administration. Database transformation playbook. https://tech.gsa.gov/playbooks/database/. Accessed: 2021-04-08.

[2] S. Alkobaisi, W. D. Bae, S. Narayanappa, and N. Debnath. Steel threads: Software engineering constructs for defining, designing and developing software system architecture. *J. Comp. Methods in Sci. and Eng.*, 12(s1):63–77, Jan. 2012.

[3] V. Averbukh, N. Averbukh, P. Vasev, I. Gvozdarev, G. Levchuk, L. Melkozerov, and I. Mikhaylov. Metaphors for software visualization systems based on virtual reality. In L. T. De Paolis and P. Bourdot, editors, *Augmented Reality, Virtual Reality, and Computer Graphics*, pages 60–70, Cham, 2019. Springer International Publishing.

[4] R. L. Bailey, C. Möstl, M. A. Reiss, A. J. Weiss, U. V. Amerstorfer, T. Amerstorfer, J. Hinterreiter, W. Magnes, and R. Leonhardt. Prediction of dst during solar minimum using in situ measurements at l5. *Space Weather*, 18(5):e2019SW002424, 2020. e2019SW002424 10.1029/2019SW002424.

[5] M. A. Bekos, B. Niedermann, and M. Nöllenburg. External labeling techniques: A taxonomy and survey. In *Computer Graphics Forum*, volume 38, pages 833–860. Wiley Online Library, 2019.

[6] H. Ben Braiek, F. Khomh, and B. Adams. The open-closed principle of modern machine learning frameworks. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 353–363, 2018.

[7] L. P. Berg and J. M. Vance. Industry use of virtual reality in product design and manufacturing: a survey. *Virtual reality*, 21(1):1–17, 2017.

[8] Cloud Native Computing Foundation. grpc.

[9] G. Developers. Ml kit. https://developers.google.com/ml-kit. Accessed: 2021-04-08.

[10] P. E. Dickson, J. E. Block, G. N. Echevarria, and K. C. Keenan. An experience-based comparison of unity and unreal for a stand-alone 3d game development course. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '17, page 70–75, New York, NY, USA, 2017. Association for Computing Machinery.

[11] C. Donalek, S. G. Djorgovski, A. Cioc, A. Wang, J. Zhang, E. Lawler, S. Yeh, A. Mahabal, M. Graham, A. Drake, et al. Immersive and collaborative data visualization using virtual reality platforms. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 609–614. IEEE, 2014.

[12] DrivenData. Magnet: Model the geomagnetic field.

[13] O. P. du Sert, S. Potvin, O. Lipp, L. Dellazizzo, M. Laurelli, R. Breton, P. Lalonde, K. Phraxayavong, K. O'Connor, J.-F. Pelletier, et al. Virtual reality therapy for refractory auditory verbal hallucinations in schizophrenia: a pilot clinical trial. *Schizophrenia research*, 197:176–181, 2018.

[14] M. El Beheiry, S. Doutreligne, C. Caporal, C. Ostertag, M. Dahan, and J.-B. Masson. Virtual reality: Beyond visualization. *Journal of Molecular Biology*, 431(7):1315–1321, 2019.

[15] Epic Games. Unreal engine.

[16] I. Fette and A. Melnikov. Rfc 6455: The websocket protocol. *IETF, December*, 41, 2011.

[17] N. C. for Environmental Information. Magnetic declination, models, data and services — ncei. https://www.ngdc.noaa.gov/geomag/, 2021. Accessed on 04/29/2021.

[18] R. J. García-Hernández, C. Anthes, M. Wiedemann, and D. Kranzlmüller. Perspectives for using virtual reality to extend visual data mining in information visualization. In *2016 IEEE Aerospace Conference*, pages 1–11. IEEE, 2016.

[19] Google. Protocol buffers - google developer.

[20] J. A. Grauer, N. H. Campbell, M. J. Acheson, and I. M. Gregory. *Dynamic Vehicle Assessment for Intelligent Contingency Management of Urban Air Mobility Vehicles*.

[21] I. M. Gregory, N. H. Campbell, N. A. Neogi, J. B. Holbrook, J. A. Grauer, B. J. Bacon, P. C. Murphy, D. D. Moerder, B. M. Simmons, M. J. Acheson, et al. Intelligent contingency management for urban air mobility. In *International Conference on Dynamic Data Driven Application Systems*, pages 22–26. Springer, 2020.

[22] T. Grubb, W. B. Garry, M. A. Brandt, T. Ames, D. C. Morton, D. Lagomasino, S. Schollaert Uz, and N. Memarsadeghi. Science data visualization in ar/vr for planetary and earth science. In *AGU Fall Meeting Abstracts*, volume 2018, pages IN53B–03, 2018.

[23] Guru99. N-tier (multi-tier) 3-tier, 2-tier architecture with examples. https://www.guru99.com/n-tier-architecture-system-concepts-tips.html. Accessed: 2021-04-08.

[24] J. K. Haas. A history of the unity game engine. 2014.

[25] IBM. Ibm cloud paks. https://www.ibm.com/cloud/paks. Accessed: 2021-04-08.

[26] E. Ivanov, A. Khoroshavin, and A. Karsakov. Visual programming environment based on data visualization grammar specification. *Procedia Computer Science*, 178:434–439, 2020. 9th International Young Scientists Conference in Computational Science, YSC2020, 05-12 September 2020.

[27] S. Kavanagh, A. Luxton-Reilly, B. Wuensche, and B. Plimmer. A systematic review of virtual reality in education. *Themes in Science and Technology Education*, 10(2):85–119, 2017.

[28] M. J. Kim, C.-K. Lee, and T. Jung. Exploring consumer behavior in virtual reality tourism using an extended stimulus-organism-response model. *Journal of Travel Research*, 59(1):69–89, 2020.

[29] M. Kirshner and D. C. Gross. Human-computer interaction for space situational awareness (ssa): Towards the ssa integrated sensor viewer (isv). In *International Conference on Human-Computer Interaction*, pages 504–515. Springer, 2019.

[30] J. Lee, R. Balachandran, Y. S. Sarkisov, M. De Stefano, A. Coelho, K. Shinde, M. J. Kim, R. Triebel, and K. Kondak. Visual-inertial telepresence for aerial manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1222–1229. IEEE, 2020.

[31] J. I. Lipton, A. J. Fay, and D. Rus. Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. *IEEE Robotics and Automation Letters*, 3(1):179–186, 2017.

[32] E. Mangortey, D. Monteiro, J. Ackley, Z. Gao, T. G. Puranik, M. Kirby, O. J. Pinon-Fischer, and D. N. Mavris. *Application of Machine Learning Techniques to Parameter Selection for Flight Risk Identification*.

[33] M. Marculescu. Introducing grpc, a new open source http/2 rpc framework. *Google Open Source Blog*, 2015.

[34] A. Miklosik and N. Evans. Impact of big data and machine learning on digital transformation in marketing: A literature review. *IEEE Access*, 8:101284–101292, 2020.

[35] P. Millais, S. L. Jones, and R. Kelly. Exploring data in virtual reality: Comparisons with 2d data visualizations. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2018.

[36] P. Millais, S. L. Jones, and R. Kelly. Exploring data in virtual reality: Comparisons with 2d data visualizations. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, page 1–6, New York, NY, USA, 2018. Association for Computing Machinery.

[37] NASA. Better than reality: Nasa scientists tap virtual reality to a make a scientific discovery. https://www.nasa.gov/feature/goddard/2020/scientists-tap-virtual-reality-for-discovery. Accessed: 2021-04-08.

[38] N. A. Neogi and J. Holbrook. Creating formal characterizations of routine contingency management in commercial aviation. In *AIAA Scitech 2021 Forum*, page 1116, 2021.

[39] E. Olshannikova, A. Ometov, Y. Koucheryavy, and T. Olsson. Visualizing big data with augmented and virtual reality: challenges and research agenda. *Journal of Big Data*, 2(1):1–27, 2015.

[40] Orange. Orange data mining - data mining. https://orangedatamining.com/. Accessed: 2021-04-08.

[41] M. Pfandler, M. Lazarovici, P. Stefan, P. Wucherer, and M. Weigl. Virtual reality-based simulators for spine surgery: a systematic review. *The Spine Journal*, 17(9):1352–1363, 2017.

[42] V. Pimentel and B. G. Nickerson. Communicating and displaying real-time data with websocket. *IEEE Internet Computing*, 16(4):45–53, 2012.

[43] J. Radianti, T. A. Majchrzak, J. Fromm, and I. Wohlgenannt. A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda.

Computers & Education, 147:103778, 2020.

[44] M. R. Rahman and S. Akhter. Real time bi-directional traffic management support system with gps and websocket. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 959–964, 2015.

[45] S. Ramachandran, M. Rosengarten, and C. Belardi. Semi-supervised machine learning for spacecraft anomaly detection diagnosis. In *2020 IEEE Aerospace Conference*, pages 1–10, 2020.

[46] A. K. Raz, E. P. Blasch, C. Guariniello, and Z. T. Mian. *An Overview of Systems Engineering Challenges for Designing AI-Enabled Aerospace Systems*.

[47] J. Rubin, D. Chisnell, and J. Spool. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. Wiley, 2011.

[48] A. Sarirete, Z. Balfagih, T. Brahimi, M. D. Lytras, and A. Visvizi. Artificial intelligence and machine learning research: towards digital transformation at a global scale, 2021.

[49] SciKit. Scikit-learn: Machine learning in python. https://scikit-learn.org/stable/. Accessed: 2021-04-08.

[50] R. Sicat, J. Li, J. Choi, M. Cordeil, W.-K. Jeong, B. Bach, and H. Pfister. Dxr: A toolkit for building immersive data visualizations. *IEEE transactions on visualization and computer graphics*, 25(1):715–725, 2018.

[51] C. Silva, W. R. Johnson, E. Solis, M. D. Patterson, and K. R. Antcliff. *VTOL Urban Air Mobility Concept Vehicles for Technology Development*.

[52] M. Sipser. Introduction to the theory of computation. *SIGACT News*, 27(1):27–29, Mar. 1996.

[53] Splunk. Cloud-based data platform for cybersecurity, it operations, and devops. https://www.splunk.com/. Accessed: 2021-04-08.

[54] Splunk. Experience your data in 3d with splunkvr. https://www.splunk.com/en_us/blog/platform/experience-your-data-in-3d-with-splunk-vr.html. Accessed: 2021-04-08.

[55] Tableau. Business intelligence and analytics software. https://www.tableau.com/. Accessed: 2021-04-08.

[56] S. K. Tadeja, P. Seshadri, and P. O. Kristensson. Exploring aerospace design in virtual reality with dimension reduction. In *AIAA Scitech 2019 Forum*, page 2206, 2019.

[57] G. Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[58] V. B. Vasudevamurt and A. Uskov. Serious game engines: Analysis and applications. In *2015 IEEE International Conference on Electro/Information Technology (EIT)*, pages 440–445, 2015.

[59] G. A. Yashin, D. Trinitatova, R. T. Agishev, R. Ibrahimov, and D. Tsetserukou. Aerovr: Virtual reality-based teleoperation with tactile feedback for aerial manipulation. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 767–772, 2019.

[60] R. Yung and C. Khoo-Lattimore. New realities: a systematic literature review on virtual reality and augmented reality in tourism research. *Current Issues in Tourism*, 22(17):2056–2081, 2019.