COMMON SOURCE & PROVENANCE AT VIRTUAL PRODUCT HOUSE

F. Dressel, German Aerospace Center (DLR), Institute of Software Methods for Product Virtualization, Germany

A. Doko, German Aerospace Center (DLR), Institute for Software Technology, Germany

In this paper we address the IT related issues which arise in the multi-fidelity, multi-disciplinary workflow of the Virtual Product House (VPH) start project. Different stakeholders are involved providing functionality as a service to simulate the end2end process from virtual design to virtual certification, including virtual manufacturing and virtual testing. Here, we present our two-fold approach for a collaborative design, implementation and operation of a common workflow among multiple stakeholders: A common simulation environment and the data tracing. A third aspect was added to support long-term reconstruction of data. We believe, that the approach presented in this paper can be used as blueprint for similar workflow settings.

1. INTRODUCTION

The Virtual Product House (VPH) is an integration and test center for the virtual simulation and certification. Within the VPH start project a continuous workflow for the virtual design, manufacturing and testing was built which allows the analysis of different movable configurations with respect to certification relevant aspects (see also: [1], [2], [3]) on an aircraft wing.

An important part of the VPH mission is the collaboration between various DLR institutes and industrial partners, which is expected to go way beyond simple knowledge exchange and involves direct interaction with the tools and data of a common workflow. To allow such a collaboration, a secure common simulation and data environment needs to be available which covers data exchange, handling of confidential data (with respect to both intellectual property and General Data Protection Regulation (GDPR)), functionality as a service and service levels. While there exists a plethora of different cloud providers which would cover the service-oriented part and also new distributed data concepts like Industrial Data Spaces [4] or Gaia-X [5], there is, to the best of our knowledge, no combined environment for aerospace workflows, which covers both aspects together. Since the environment proposed in this paper allows collaboration and at the same time protects intellectual property it is positioned between the open source and proprietary solution approach. Therefore, we call it Common Source. In the current version of the VPH start project workflow, there is collaboration between different DLR institutes and one external university. This allows the demonstration of the Common Source environment.

In 2020 the European Union Aviation Safety Agency (EASA) released a proposal for a certification memorandum regarding modelling and simulation [6]. It defines criteria for using simulation to support certification tasks. This defines a strong fundament for certifiable aerospace workflows from an aviation domain point of view. It describes the domain requirements when using simulation for certification as well as some basic software engineering guidelines.

Within the VPH start project workflow, multiple disciplines are coupled and multiple stakeholders are involved. This adds another level of complexity compared to using only solitary tools. The complexity can only be handled, if the data flowing through the workflow and between the stakeholders are thoroughly tracked. This can be done with provenance (see [7] for an overview), which is a special kind of metadata about the origin of data. It helps to understand data and the corresponding data flow. Tracking provenance is common in scientific workflows (see for example [8]) and a lot of data exchange formats are in some way provenance-aware or can at least store provenance. The multiple stakeholder setup of the VPH workflow poses new challenges with respect to data tracing due to its distributed nature and tools which are only available via interfaces defined within the Common Source framework.

In the following sections, we explain the workflow we used as the underlying use case, the Common Source approach and the measures we took to trace data with provenance.

2. THE USE CASE

The VPH start project workflow is a multi-fidelity, multi-disciplinary workflow for virtual design, manufacturing and testing. It starts with configuration files in which the wing

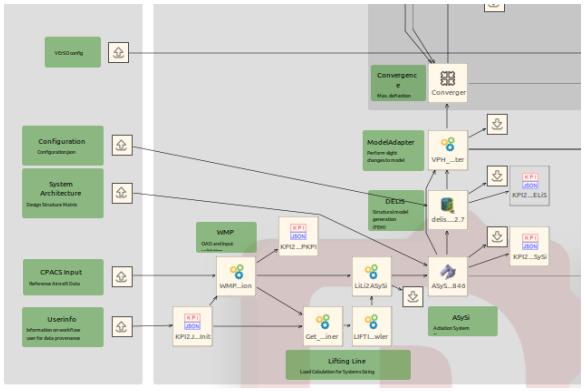

Figure 1: Excerpt of the VPH workflow from the workflow orchestration tool. Shown are the first tools of the workflow (big boxes) as well as the workflow input (left).

moveable configuration and load cases are defined. The workflow consists of tools provided as black box by different stakeholders (both DLR internal and external) which cover the steps from design to testing. The resulting data should be used for virtual certification.

The tools provided by the different stakeholders are executed on distributed machines for which the internals like soft- and hardware as well as the operating system are hidden to users. Files and directories are used for data exchange. The tools are managed by a workflow orchestration tool, which takes care of the data transfer (via files and directories) and the execution of the tools in the right order. See Figure 1 for an excerpt of the VPH workflow.

3. COMMON SOURCE

3.1. Related work

When it comes to concepts or solutions that allow collaboration but at the same time protect intellectual property several large-scale concepts should be mentioned. The first and most famous example is Gaia-X – an infrastructure and data ecosystem following European values and standards [5]. It can be described as an open cloud with well-established standards, self-descriptions, policies, identities and trust. This vision also includes collaboration but with ensuring data sovereignty of each participant which is the main topic of Common Source.

International Data Spaces is another approach which could be positioned above Gaia-X with the aim to foster integration of data in the future digital economy [5]. Again, one of the main topics is the control of data by the owner (data sovereignty).

Specialized collaboration software like PLM (Product Lifecycle Management) is also related which allows collaboration during the whole product life cycle or specialized solutions for aerospace industry like BoostAerospace [9]. Such kind of solutions also allow some degree of data sovereignty protection. It has the disadvantage that all the stakeholders have to agree to use one software incurring additional organizational costs. PLM is mostly used within one company.

We used our own architecture (Remote Component Environment, RCE [10]) for the VPH workflow, which is less distributed than a cloud (the servers are explicitly provided by the stakeholders) but more flexible than PLM or centralized data management systems while still keeping well defined interfaces for data exchange.

3.2. Basic idea behind Common Source

The base concept of Common Source should be demonstrated on an example. Imagine a big aircraft manufacturer who wants to develop an airplane part in collaboration with subcontractors. In Figure 2 four stakeholders are shown: main company *A* which initiated the development at the top and three subcontractors (*B*, *C*, *D*). With Common Source company *A* shares its unfinished version of the aircraft part with subcontractors (*a*). Subcontractor *B* has his own tools which it uses for development of aircraft parts. These tools can be integrated into a common workflow without sharing of source code using a black box approach which will be explained later. These tools can be executed by the workflow at the site of partner *B* (*b*). Data from partner *C* is used as input for further processing (*c*). After the workflow is done, the result is immediately available to the main company (*e*). During the whole development process (i.e. during the execution of the common workflow) data like results of single steps or provenance information is collected to be immediately available to all stakeholders. This allows for example an online health check of intermediate results. Everybody is capable to analyze the data (*d*) and draw conclusions which can be used for further improvements as shown with partner *D*.

Based on the example given, we define the Common Source as following:

A concept for collaborative virtual product development in a decentralized work environment with multiple partners and special emphasis on data sovereignty.

3.3. Components of the Common Source

To implement Common Source, different components and tools need to be combined. For the VPH-workflow, the following was chosen. First there are the participants. In the Ecomat building in Bremen a common plateau is created, where meetings between the stakeholders were possible.

While the people are connected via direct (and indirect) meeting options, the tools and data need to be plumbed together with a workflow orchestration tool. We have chosen RCE [10] as it allows providing tool functionality for others without disclosing the source or binary code and defines simple but effective interfaces between the tools. The workflow orchestration, tool execution and data exchange are completely triggered by RCE.

The tools are hidden between the stakeholders in Common Source but the data is not. Therefore, tooling is needed to easily access and evaluate the data of each workflow run. While RCE transports the data while the workflow is running, we tested different (centralized) storage concepts like git lfs [11], file shares, SharePoint [12] or data management systems. File shares seem to work best. This is mainly due to its ease of use and the correspondence to the data exchange in the VPH workflow, which is done via files. File shares and their connection to data management systems will be followed on in the future. The choice also affects the way, provenance information is available to the stakeholders. In the current implementation of the workflow, provenance is available at the interfaces of the tools in a

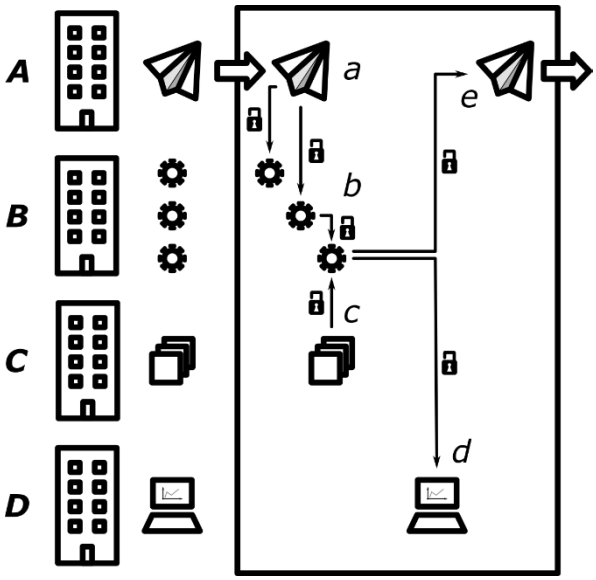

Figure 2: Exemplary setup of a Common Source project, in which stakeholder A – D collaborate to enhance the design of an aircraft part.

file-based form. This can be stored along the data in a file share or be indexed and made searchable in a data management system.

To enhance the whole process, additional components like code management systems, continuous integration tooling or messaging is needed. We have chosen git, GitLab CI [13] and Mattermost [14] respectively.

3.4. Integration of tools into RCE workflows and black box approach

RCE is a workflow orchestration tool developed at DLR. It allows multiple partners to share tool execution and data in one common workflow without disclosure of source or binary code.

To integrate a tool (e.g. calculation, simulation) into an RCE workflow it has to meet the following requirements:

- The external tool must be callable via command line in Linux or Windows
- It must have a non-interactive mode
- Input for the tool must be provided through command line arguments or files

RCE provides unified interfaces between the tools with the following data types:

- File
- Directory
- ShortText
- Float
- Integer
- Boolean

To integrate a tool, the tool provider has to provide a server with RCE installed on it, on which the tool is running. It is important to emphasize that each tool resides on the owner's site (i.e. it executes on the owner's server behind a firewall). Only inputs and outputs are exchanged. We call this approach black box because neither the source code nor the executable file are accessible to partners.

3.5. RCE's Uplink feature

To allow integration of distributed tools into RCE workflows, networks of two or more partners have to be connected over the internet in a secure manner. The feature, which is responsible for secure communication between partners in RCE, is named Uplink. It is based on the SSH network protocol which provides well-tested encryption and login authorization. As additional security feature a relay server is used that can be placed outside the organization's internal network which eliminates the need to open any incoming network ports in the organization's firewalls. Besides a relay server, a typical configuration involves an Uplink SSH gateway node in each organization's network which is the only node that actually establishes an SSH connection to the relay. A gateway node simplifies the configuration of the relay since it reduces the number of SSH logins that must be configured on it. Once the setup is done, users from different organizations can publish their own tool services via the Uplink relay or use the ones published by others. Figure 3 shows an RCE network with different configuration setups possible with Uplink.

3.6. Connection of external partners to the Common Source

At the moment the University of Bremen is connected to the Common Source infrastructure as external partner alongside the DLR institutes involved in the VPH start project. It is the prototype user to test connections also to other stakeholders outside the DLR. We established a connection using RCE's Uplink feature. An account was set up with username and password for the University of Bremen for connecting to Uplink relay using RCE. University of Bremen was able to execute the distributed VPH workflow together with DLR institutes. They contributed a tool called KPI2JSON to the workflow. It allows the saving of key performance indicators (KPI) of the workflow to a central JSON file. The json file is transported and enriched throughout the complete data flow and thus it always represents the current state of the workflow with respect to the domain KPIs. The tool can be considered an addon to existing assessment tools from the DLR Institutes. One should emphasize the basic feature of the Common Source – data sovereignty of the tool: Even if the University of Bremen did publish the KPI2JSON tool it did not reveal its source code nor the executable. The execution of the KPI2JSON tool takes place on servers of the University of Bremen.

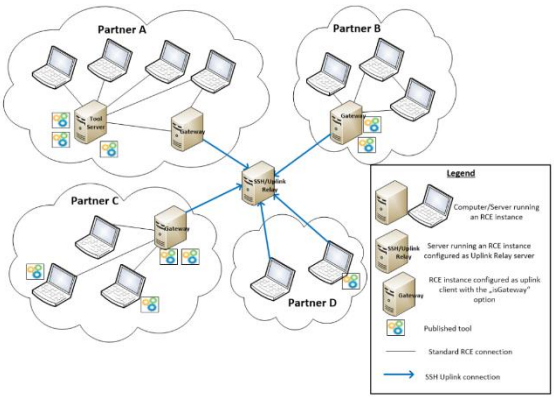

Figure 3: Example RCE Network with various Uplink configurations

4. PROVENANCE TRACKING

Provenance is a special kind of metadata describing the origin of data. It contains data such as author of a data set, tools with which it was created or creation/modification timestamps, etc. Provenance is essential to understand the flow of data in a retrospective manner as is the case in virtual certification and data reuse tasks.

Provenance can be recorded for many different use cases (see again [7] for a survey). The use case determines the level of granularity needed (see [15] for an example with very fine grained provenance). There are ways to record provenance prospectively by source code analysis (see for example [16]) or retrospectively by recording the steps which were done while processing the data (see for example Arvados as a workflow engine supporting provenance [8]).

Based on our use case, an end2end virtual process for a certifiable wing movable, we decided to use a coarse-grained provenance approach with retrospective provenance to provide enough information about the data and the data generating process to allow data reuse over the lifetime (may be up to decades) of a (virtual) product.

4.1. Assumptions and requirements

We made some assumptions within the context of the use case, which drove the technical implementation. The first three assumptions are pessimistic and reflect the inescapable decay of software systems. While the decay can be stopped with sufficient resources, the costs increase over time and it is implicitly assumed here, that the stakeholders reusing the data are either not willing to bear the costs or are not in the position to do so.

The assumptions were the following:

Tools (in a given version) may not be runnable anymore at the timepoint of data reusage. While this is not an issue during development of the workflow and its productive use, it is an issue when trying to reuse the data for decades. For example, a study on Java libraries found a high rate of breaking api changes especially in popular libraries which even increases with the lifetime of the library [17]. Even if the tool is available, it may be too much effort to setup the surrounding interfaces and systems to run it in the same version and environment again.

The workflow is not runnable anymore at the timepoint of data reusage. The same considerations applicable to a single tool apply to the workflow in total as a software defined process. It is even worse because changes in tools, while preserving the functionality, may change their input/output formats and render the interfaces in the workflow useless.

Hardware is not available anymore. There are two aspects to this assumption: First: The hardware for running the workflow is physically not available anymore or is too expensive to setup again. Second: Some tools used some specific hardware-targeting implementation which does not work anymore on hardware available at the timepoint of the data reusage.

Data is kept in some suitable storage. Based on the current cheap and efficient storage for data (see [18] for exemplary data from a cloud provider) we assume that data will safely travel through time. Established concepts for error detection and correction exists. There is the threat, that tools for reading the data in the given format are not available anymore. This can be mitigated by using only such common standards, for which multiple implementations exists and which are widely used. The data kept should also contain the test data which is needed to evaluate the tools and workflows and the corresponding documentation for the tools. We do not assume the existence of a common data management system and therefore refer only to simple data storage.

Provenance, data and documentation is enough for an experienced person to understand and evaluate the results of a workflow. With well tested and validated tools (and the availability of the corresponding test data), it should be possible to evaluate the behavior of the used tools. The information stored in provenance should provide enough information to understand the data flow between the different tools and thus within the whole workflow.

Based on the assumptions and the use case, the following requirements were identified:

Deal with arbitrary (binary) data to support wide range of existing and future tools. Each stakeholder in the VPH workflow can add new tools. There should be no constraint on the data consumed and produced for these tools beside being files.

Provenance and data are immutable. The Common Source concept allows to share input/output data and tools

between different stakeholders. There is no data management system in place to track changes to data and/or provenance. Therefore, immutability is the only way to ensure consistency of processing steps along the chain.

Provenance and data need to be stored in a simple way. Because the data should be usable for a long period of time, it should be possible to read and interpret provenance and data with standard tools without sophisticated programming experience.

Human readable standards for provenance information should be used. Human readability is the key for understanding and thus reusing the data. As for the points above, the existing standard increases the chance of having a living data format at the timepoint of data reusage.

The implementation needs to be able to change and evolve. It is expected, that during the productive usage of the VPH workflow new tools and approaches emerge which make changes necessary. This should be possible without breaking existing data.

Data, the corresponding provenance and each reference is uniquely and unambiguously identified. Because there is not necessarily a common data management system, each digital artefact needs a unique id. References between artefacts should be done using this id.

4.2. Concept & Implementation

Based on the assumptions and requirements, we designed a data container format which combines data and provenance in one file. The layout is shown in Figure 4. The file starts with a 24 byte long header section, followed by a section with utf-8 encoded provenance information followed by the original data.

The header section is divided into two parts: A 14 byte long format identifier and a 10 byte long representation of the number of bytes used for the provenance section; both as utf-8 encoded text. The number of bytes for both parts is somewhat arbitrary but expected to be enough to store meaningful format identifiers (for example in the format YYYYMMDDhhmmss) and sufficiently high numbers for specifying the number of provenance bytes.

The provenance section stores a simplified W3C Prov [19] model (see Figure 5) in json-representation as utf-8 encoded text. Only one entity per container is allowed. A placeholder value is used for the entity id. While using the provenance information, it is replaced with the filename of the container itself (see below). This allows using the container equivalent to an entity in the W3C model. Based on the W3C model, the provenance contains references to other used entities by entity id.

The data section contains the original data without any modifications.

As filename for such a container file, the hash over the whole file is used. This allows an easy detection of changes which were done after the file was published and on the other hand allows for a unique identifier for each container file and thus each entity (see above). With publication, the process of distributing the container is meant. The location needs to be readable by others but not modifiable by the container creator. This is for example the case when sending the container via email or storing it in a content addressable storage. We used sha256 [20] to calculate the hashes.

Reading the provenance information from such a file is as simple as:

- Stripping away the first 14 bytes and make sure, it corresponds to the current version (may determine hash function for filename, …)
- Convert the next 10 bytes to a number
- Read as many bytes as determined by the previous step from the file while skipping the first 24
- Replace entity id placeholder with the hash value of the container content (filename)

Reading the data from such a file is as simple as:

- Stripping away the first 14 bytes and make sure, it corresponds to the current version (may determine hash function for filename, …)
- Convert the next 10 bytes to a number
- Skip the number of provenance bytes, determined in the step before
- Read the remaining file

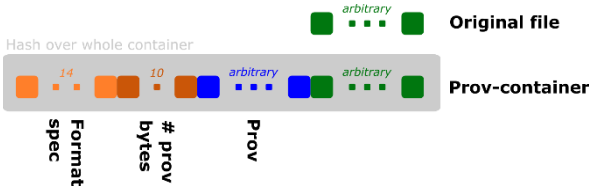

Figure 4: Byte layout of the provenance container. See text for details.

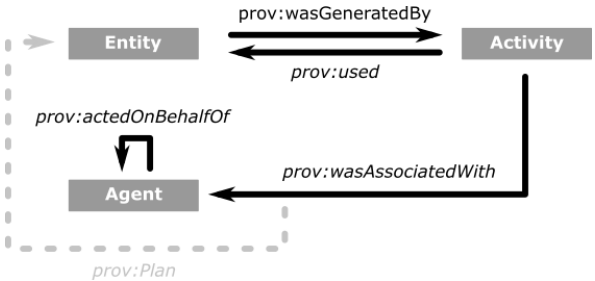

Figure 5: Simplified provenance model used for annotating the VPH workflow data with provenance.

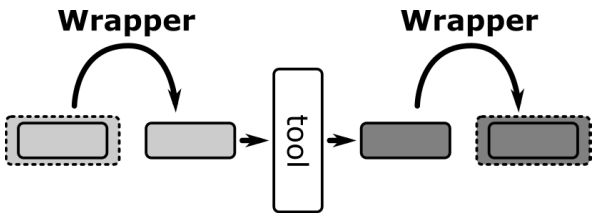

Figure 6: Pre- and post-processing of container before passing them to/from the corresponding tool.

We created a python library to easily deal with the container format described above. While reading and writing of the container format is simple and can be done even with a command line one-liner, the main focus of the library is additional support for gathering of provenance and further analysis of the containers. To populate the provenance section of each container, some information is gathered by the library (dynamic information) and some is provided by the tool creator or the workflow orchestration tool. The dynamic attributes are:

- Current timestamps
- Input-dependencies (every container in input directory)
- Output information (filename, creation date, …)
- W3C specific provenance information like ids, associations between agents and entities, …

We used RCE as workflow orchestration tool for the VPH workflow. The data transfer within the workflow was done via file/directory exchange within RCE. It allows the integration of arbitrary tools for which pre- and postprocessing steps can be defined. We used this mechanism to convert input data files from the container format before each tool was run and convert results back to container format files after the tool was run. This keeps the data exchange (file based) completely unaffected. See Figure 6 for a visual representation.

The provenance information with the simplified model shown in Figure 5 describes a graph, which can be used to gather knowledge about the workflow run. Figure 7 shows as an example the provenance graph of the virtual testing subbranch of the VPH workflow. It shows the relation between the results of the model generation, orange points in bottom layer, to the green and blue labeled parameter files which were used as inputs and are additionally labelled with the corresponding git commits as part of their provenance information.

4.3. Comparison to related work

Data container formats for enriching data with additional information are well established. This starts with classical formats from photography like jpeg and continues with data formats used in science and engineering like hdf5 [21], FITS [22] or netCDF [23]. Newer approaches are Research Object Crates [24] or frictionless data [25]. Nevertheless, none of these formats were able to fulfill all our requirements for provenance. The main reasons were first a too high complexity (compared to our simple approach), second the mutability of data (which was allowed for example by using non-content-addressable references to other data) and third non-standard metadata schemes (not using the de facto standard W3C Prov model).

The approach presented here allows FAIR [26] data when combined with additional data management systems. While the container format is interoperable (using W3C prov model and corresponding attributes) and reusable (providing enough provenance to reconstruct the workflow after long periods of time), the findability and accessibility aspect of FAIR need to be provided by external systems.

5. TRUST

Although provenance provides information about the original data flow it is not enough to reconstruct data after a long period of time. We derived some additional requirements on workflow and tool level which should make a reconstruction after decades possible. It extends and build upon the provenance approach presented above. It allows to estimate the amount of information needed to provide with provenance and to check some underlying assumptions with respect to long-term data reusage. In the VPH, this is used as additional considerations for integrating tools in the workflow.

TRUst is a combination of three aspects: **T**estable and tested, **R**econstructable and reconstructed and **U**nder**st**andable and understood. It overlaps with software engineering quality requirements and also with requirements regarding virtual certification (see for example [6]), which also favors testing and test-driven development. Nevertheless, the focus of TRUst is data understanding and reusage and not software quality. The concept is bimodal with an active and passive part for each aspect: Tools need to be testable but the tests need also to be performed (regularly), the tools and workflow need to be

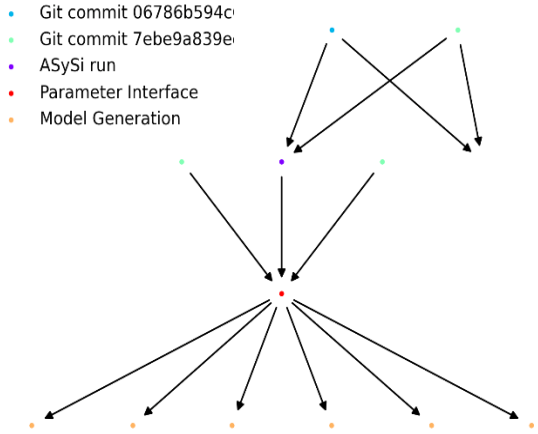

Figure 7: Provenance graph for the data generated with the data from the virtual testing branch. See text for details.

Testable & tested
Version control
Technical test data and domain test data available
Automatic tests
Tool/workflow output comparable to test data
Reconstructable & reconstructed
Exemplary data available, usage documented
Installation on different hardware simple for experts
Tools can be exchanged, interfaces simple
Understandable & understood
Documentation for tools and workflow available
Long term storage of workflow data
Provenance of each critical artefact
Simple input/process/output model for tools

Table 1: Tasks for TRUst.

reconstructable in different environments but also need to be reconstructed, the same holds for understanding.

In the VPH, we used the tasks according to Table 1 for each aspect.

While the test aspect is mainly addressed on tool level by each tool provider, the reconstruction tasks are connected to the Common Source concept and the understanding tasks relate to provenance and the requirements we defined above as well as to the Common Source concept.

6. CONCLUSION

In this paper we presented the key ideas behind the IT concept for the VPH start project workflow. It consists of two major parts: Common Source to orchestrate a workflow of tools and a provenance data container format to track the generated data for long-term data reusage.

Common source is a holistic concept for collaborative development of virtual products with special emphasis on data sovereignty (by using a black box approach). We presented the main elements of it including the RCE which is a collaborative workflow orchestration tool with a black box approach to execute distributed tools. We presented RCE's Uplink feature, which allows save communication between distributed organizations. A first use case with the external partner University of Bremen was shown.

We proposed a simple data container format which stores data and provenance together. It is stripped of anything not strictly necessary for the workflow goal of providing data for virtual certification or which can be provided by other means. This especially means that these containers are not meant to be FAIR [26] by themselves. Nevertheless, the proposed container format can be made fair by external data management systems (providing search functionality, licensing, creating links, …). Data and provenance within the container are strictly immutable. The immutability is enforced by using the hash over the container content as filename/entity id. As soon as such a container is distributed to other stakeholders (as done within the Common Source environment), a later modification to either provenance or data will be detectable by comparing the hash of a container with the information stored in provenance for previous steps of the workflow.

The proposed data container format fits perfectly into the scenario of distributed stakeholders connected loosely by the Common Source environment. It can transport the information needed to understand and assess data from the virtual design, virtual manufacturing and virtual testing steps of the workflow and thus enables the usage of the data for virtual certification. To further enable long-term data reuse additional requirements (TRUst) for the workflow design were set up, which built on the Common Source and provenance approach described above.

The approach presented here is not specific to the VPH start project. The Common Source approach is general enough to be used for other settings with multiple stakeholders. It is useful every time, where the stakeholders show enough commitment to share resources (tool functions, tool infrastructure and maintenance) while keeping the tools themselves hidden. The black box approach allows each stakeholder to keep their intellectual property safe. On the other hand, this makes it harder for the other stakeholders to ensure the functionality of the corresponding tools which is needed for quality control during runtime and for reconstruction of data after long time. TRUst defines additional supporting measures and enables for example functional duck typing based on its focus on testing and common test data, continuous integration and others. The provenance container format can be applied whenever a data exchange is needed which should preserve provenance. The format supports immutability of data and provenance inherently and is simple enough to keep the maintenance effort for the data storage and data reuse small even in the long term. If needed, data management systems can parse the format and add attributes to support more complex use cases and make the provenance container FAIR.

7. ACKNOWLEDGEMENTS

The Virtual Product House start-up project is funded by the German federal state of Bremen and the European Regional Development Fund (ERDF).

European Union
Investing in Bremen's Future
European Regional
Development Fund

8. REFERENCES

[1] Hollmann R, Schäfer A, Bertram O, Rädel M. Virtual Testing of Multifunctional Moveable Actuation Systems at Virtual Product House. In: Deutscher Luft- und Raumfahrtkongress, 2021.

[2] Lange F, Zakrzewski AS, Rädel M, Hollmann RW, Risse K. Digital Multi-Disciplinary Design Process for Moveables at Virtual Product House. In: Deutscher Luft- und Raumfahrtkongress, 2021.

[3] Rädel M, Delisle D, Bertling D, Hein R, Hollmann R, Lange F et al. Towards Robustness Assessment in Virtual Testing - Manufacturing Influences by Simulation-based Methods in the Virtual Product House. In: Deutscher Luft- und Raumfahrtkongress, 2021.

[4] Otto B, Auer S, Cirullies J, Jürjens J, Menz N, Schon J et al. Industrial Data Space: Digital Sovereignity over Data. [August 03, 2021]; Available from: https://www.fraunhofer.de/content/dam/zv/de/Forschungsfelder/industrial-data-space/Industrial-Data-Space_whitepaper.pdf.

[5] DE-CIX Management GmbH, Eggers G, Fondermann B, Google Germany GmbH, Maier B, Ottradovetz K et al. GAIA-X: Technical Architecture. [August 03, 2021]; Available from: https://www.bmwi.de/Redaktion/EN/Publikationen/gaia-x-technical-architecture.pdf.

[6] EASA. Notification of a Proposal to issue a Certification Memorandum: Modelling & Simulation – CS-25 Structural Certification Specifications. [August 03, 2021]; Available from: https://www.easa.europa.eu/sites/default/files/dfu/proposed_cm-s-014_modelling_simulation_-_for_consultation.pdf.

[7] Herschel M, Diestelkämper R, Ben Lahmar H. A survey on provenance: What for? What form? What from? The VLDB Journal 2017;26(6):881–906. https://doi.org/10.1007/s00778-017-0486-1.

[8] Arvados developer. Arvados. [February 23, 2021]; Available from: https://github.com/arvados/arvados.

[9] BoostAeroSpace. BoostAerospace. [July 19, 2021]; Available from: https://boostaerospace.com/.

[10] Boden B, Flink J, Först N, Mischke R, Schaffert K, Weinert A et al. RCE: An Integration Environment for Engineering and Science. SoftwareX 2021;15:100759. https://doi.org/10.1016/j.softx.2021.100759.

[11] Git LFS contributors. Git Large File Storage. [August 04, 2021]; Available from: https://git-lfs.github.com/.

[12] Microsoft Corporation. SharePoint. [August 04, 2021]; Available from: https://www.microsoft.com/en-gb/microsoft-365/sharepoint/collaboration.

[13] GitLab Inc. GitLab. [August 03, 2021]; Available from: https://docs.gitlab.com/.

[14] Mattermost Inc. Mattermost. [August 03, 2021]; Available from: https://mattermost.com/.

[15] Sultana S, Bertino E. A file provenance system. In: CODASPY '13: Proceedings of the third ACM conference on Data and application security and privacy, 2013.

[16] Namaki MH, Floratou A, Psallidas F, Krishnan S, Agrawal A, Wu Y et al. Vamsa: Automated Provenance Tracking in Data Science Scripts. In: KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020.

[17] Xavier L, Brito A, Hora A, Valente MT. Historical and impact analysis of API breaking changes: A large-scale study. In: KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020.

[18] Andy Klein. Backblaze Hard Drive Stats for 2019. [July 07, 2021]; Available from: https://www.backblaze.com/blog/hard-drive-stats-for-2019/.

[19] Moreau L, Missier P. PROV-DM: The PROV Data Model. [February 23, 2021]; Available from: https://www.w3.org/TR/prov-dm/.

[20] Dang QH. Secure Hash Standard. National Institute of Standards and Technology; 2015.

[21] Koziol Q, Robinson D. HDF5. Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA (United States); 2018.

[22] Wells DC, Greisen EW, Harten RH. FITS - a Flexible Image Transport System. Astronomy and Astrophysics Supplement Series 1981;44:363.

[23] Rew R, Davis G, Emmerson S, Cormack C, Caron J, Pincus R et al. Unidata NetCDF. UCAR/NCAR - Unidata; 1989.

[24] RO-Crate contributors. RO-Crate Metadata Specification 1.1. [June 30, 2021]; Available from: https://www.researchobject.org/ro-crate/1.1/.

[25] Frictionless Specifications contributors. Frictionless Specifications. [February 23, 2021]; Available from: https://github.com/frictionlessdata/specs.

[26] Wilkinson MD, Dumontier M, Aalbersberg IJJ, Appleton G, Axton M, Baak A et al. The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 2016;3(1):160018. https://doi.org/10.1038/sdata.2016.18.