DATA MANAGEMENT FOR QUALITY ASSURANCE IN SEMI-AUTOMATED PROCESSES

M. Vistein, D. Nieberl, P. Kaufmann, A. Buchheim
German Aerospace Center (DLR), Institute of Structures and Design,
Am Technologiezentrum 4, 86159 Augsburg, Germany

Abstract

Data acquisition, storage and evaluation is a key requirement for quality assurance in the aircraft industry. In this paper, an approach using the software “shepard” is shown. The exemplary process contains of many manual or semi-automated steps; therefore a graphical user interface has been developed that assists the user in providing all necessary data and linking it with other data that can be acquired automatically. The practicability of this approach has been evaluated within the manufacturing process of a carbon-fiber reinforced plastics part for a helicopter.

Keywords

data management; quality assurance; automation; graphical user interface

NOMENCLATURE

Abbreviations

API	Application Programming Interface
CFRP	Carbon-Fiber Reinforced Polymer
JSON	JavaScript Object Notation
MES	Manufacturing Execution System
REST	Representational State Transfer
SCADA	Supervisory Control and Data Acquisition

1. INTRODUCTION

In aircraft industry, the certification of components is a crucial part of production. Quality assurance measures as well as information on materials and methods need to be monitored and tracked. Therefore, large amounts of data need to be captured, stored and referenced in order to fulfill admission criteria. As part of the project “NEUTRON”, data acquisition and -management based on the software tool “shepard” has been evaluated.
The “shepard” (A **s**torage for **he**terogeneous **p**roduct **a**nd **r**esearch **d**ata) software stack has been developed at the Institute for Structures and Design and aims at providing an integrated system for storing and linking all kinds of manufacturing data. This includes automatically acquired continuous data stored in time series (e.g. temperature) as well as discontinuous data like data sheets, photographs, etc. In order to access all data, a hierarchical representation of the product and process is formed using virtual data objects, and additional data like time series or binaries are referenced to allow for a precise attribution.

Shepard itself provides an application programming interface (API) and a generic web-based interface. The API can be used by automated processes to access all data stored in shepard. If a process is completely automated and centrally controlled (usually using a manufacturing execution system), this is an easy way to gather all incoming data. Using shepard in an almost completely automated process has been demonstrated in [1]. Similar to industrial production lines, the NEUTRON project consists of a large amount of process steps that are a mixture of automated, partly automated and even completely manual processes. While the acquisition of continuous data can still be automated, many additional data can only be provided by the process experts during manufacturing.
To facilitate this process, as part of NEUTRON, a graphical user interface (GUI) has been developed. This GUI allows the process experts first to design the overall hierarchical structure of the process using a graph-based interface. An important part of this step is the definition of required data, e.g. data sheets, inline quality assurance measures, NDT data or facility configurations. The goal is to provide both traceability for validation and verification purposes, as well as a basis for future improvements of the process.
Once the process has been defined, the relating structure is created in shepard. To support the manufacturing team, a wizard like GUI is presented that asks for all relevant manual input. By tracking the start and end times of each process step, also an automatic reference to continuously retrieved data is created.
Section 2 provides more technical details of the shepard data management system, and in section 3 the new *shepard Process Wizard* is introduced. In section 4 the *shepard Process Wizard* is put into context within the project NEUTRON are shared in section 4, and the paper is concluded in section 5.

doi: 10.25967/650226

2. SHEPARD

The shepard project is being developed at the Institute of Structures and Design at the German Aerospace Center (DLR) and aims at providing an integrated system for storing and linking all kinds of manufacturing data [2, 3]. There are several key concepts in shepard which are used to structure the data:

- **Collections** are used to group data on the top level and represent e.g. a project, a product, etc.
- **DataObjects** are the basic organizational element in shepard. Every DataObject belongs to exactly one Collection, and DataObjects can have relations (parent/child, predecessor/successor) to other DataObjects.
- **Containers** store the real data. There are containers for time series (data values stored with a timestamp), arbitrary binary files or structured data (generic data that can be represented as a JSON object).
- **References** link from DataObjects to data stored, e.g. to a certain file stored in a file container, to a time span in a time series, etc.

The basic interface to shepard is a REST API. Custom applications can use it to store and retrieve data from/to shepard, and to create all references as necessary. Shepard provides automatically generated client implementations for some commonly used programming languages, and further client implementations can be generated using the OpenAPI project.
Besides the API, a web-based frontend is provided that allows the user to perform many basic operations such as creating or deleting elements.

3. SHEPARD PROCESS WIZARD

If a production process can be performed completely automatic, the integration of shepard as data storage is comparably simple. The process is usually controlled centrally by a SCADA or MES system. These systems can be extended with an interface to the shepard API in a way that the required data structures (Collections and DataObjects) are created on-the-fly as needed. Sensor systems can be automatically configured to provide measurements as time series and references with appropriate start and end times can be generated by central controller.
In research applications – but also in some production environments – not all steps can necessarily be fully automated. In a mixture of automated and manual process steps, no single central control unit is available that could be responsible for maintaining the data structures in shepard.
Manually creating a data structure in shepard – either using the API or the web frontend – is a tedious task, in particular if the same structure is needed several times, e.g. for multiple tasks. Therefore, the *shepard Process Wizard* has been created which provides a graphical user interface for the process definition as well as for the later (multiple) process executions [4].
In section 3.1 the application for process definition is described, and in section 3.2 the process execution.

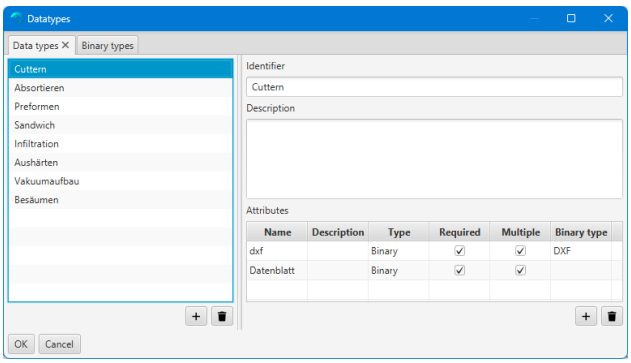

FIG 1. Graphical editor for definition of process step types and the required data for each type.

3.1. Process Definition

In the process definition step, a template for the process is created, which then can be applied for each part that is produced. In order to define the process, two independent definitions must be done:

1) Which data must be acquired in each type of process step (see section 3.1.1)
2) Which process steps are necessary, and in which order (see section 3.1.2)

Both definitions have been separated, because it is often required to record the same set of data for different steps of a production process, therefore the definition of required data can be reused.

3.1.1. Required Data

In order to model a process, types of process steps need to be identified. A type of process step in the *shepard Process Wizard* is defined by the set of data that needs to be acquired, stored or referenced for a certain process step. While it is possible that every process step has its own type, usually several process steps can share a common type. The definition of required data can be done using a graphical editor as seen in fig. 1.
For each process step type, a number of data items can be specified. For each item specified, the user will be asked to provide input later during the process execution (see section 3.2). For each item, a concrete data type must be specified. Besides common types such as `String`, `Integer` or `Float`, also binaries are possible (e.g. to require the user to upload a PDF file). Furthermore, also time series ca be selected as data type which will allow the user to link the execution of a certain process step with a set of data automatically captured in a time series. Each item can have a flag marking it as required which will be highlighted during execution. It is also possible to allow an item being used multiple times, e.g. if the user should be allowed to upload multiple documents for a certain item.

3.1.2. Process Flow

In order to represent the logical structure of a production process, DataObjects are used in shepard. DataObjects can be linked to each other using parent/child and predecessor/successor relations. Shepard itself handles these

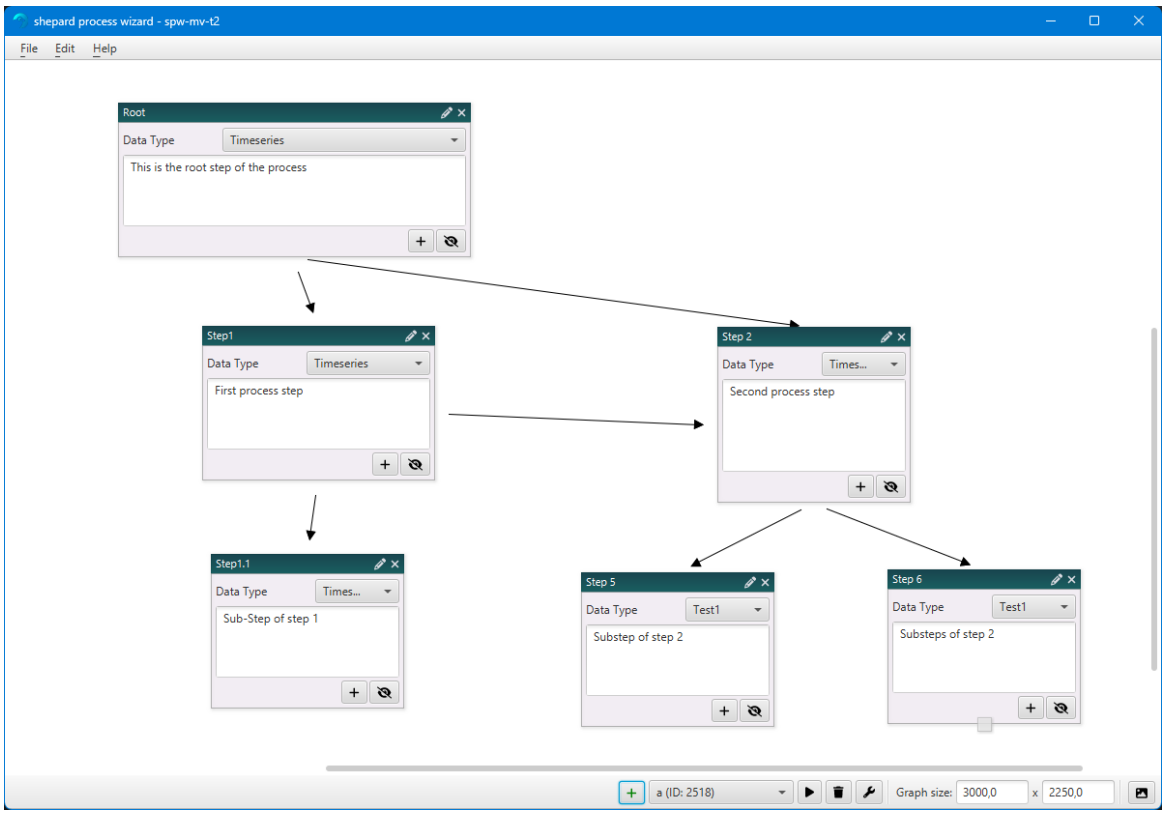

FIG 2. Graphical editor for modeling of process flow.

relations as a directed graph and does not place any constraints on these relations, i.e. cyclic connections are allowed. For the *shepard Process Wizard* however it has been decided to restrict the modeling to an acyclic directed graph with a single root node (a tree-like structure).

The *shepard Process Wizard* provides a graphical editor for modeling of the process flow (see fig. 2). Each process step is represented as a box which can be freely arranged on the work space. Connections can be created by dragging lines between the boxes. Two types of connections can be created:

- **Parent/child** relations are connecting from the top and bottom sides of the process step box and create a hierarchical order of process steps. There must be a single root node which has no parent that represents the whole process. Children can be e.g. different parts of which a work piece is made of, or different sequential sub-process steps.
- **Predecessor/successor** relations are connecting at the left and right side of the process step boxes. These relations define the order of executions for multiple process steps that are on the same level, hence only process steps that have the same parent step can be connected using these relations. If two process steps have the same parent but no predecessor/successor relation, the order of execution of both steps is undefined. Cyclic relations are not allowed.

For each process step, a process step type as defined in section 3.1.1 must be selected. This defines which data is collected for each process step. Every process step may collect data, not only the leaf steps. However, it is also possible to define step types that do not collect any data (e.g. for the high-level steps).

3.2. Process Execution

The model that is created as described in section 3.1 is not immediately mapped to a shepard structure. While the model is still under development, it is stored as structured data consisting of two JSON documents. One is representing the logical structure of the process, the other one the graphical aspects (position of the boxes, connections, etc.). Storing the structure directly by means of DataObjects would be unnecessarily difficult to maintain.

The execution of a process is done in two steps:

1) Creating all necessary structures within shepard to represent the model created.
2) Guiding the user through the process and supporting the provision of necessary process data.

3.2.1. Initial Creation of Structures

In order to start with the execution of a process, an instance (called *run*) of the model must be created. In a first step, the validity of the model is checked. While the graphical editor prevents the user from creating many invalid connections such as cycles, not all properties can be enforced there, e.g. the requirement for a single root node can only be checked during instantiation.

Once the validation was successful, the model is transferred to shepard. Every process step in the model is represented by a DataObject, and the relations are created using the built-in features of shepard for linking DataObjects. At the moment, no further changes to the model

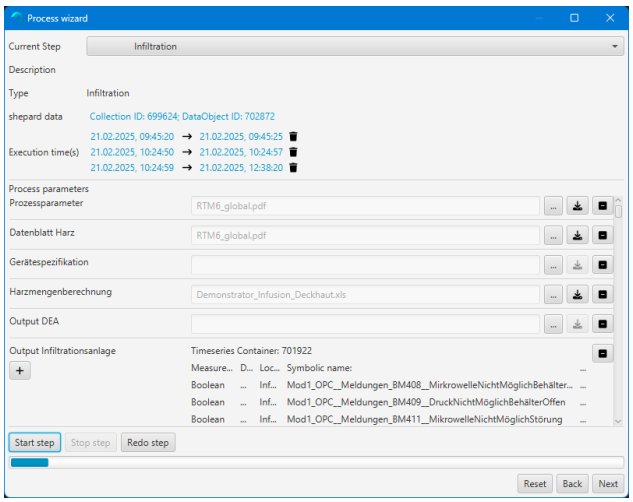

FIG 3. Graphical user interface for providing necessary process data.

are possible once a run has been created. Allowing modifications would require to track the modifications the user performs and trying to apply them to the existing structures. While this might be easily possible for some changes such as simply adding new steps, this can be tricky for deletions or other modifications. If the run has already been filled with data, this could even lead to data loss when a node is inadvertently deleted. Creating a new run with a modified model however is always possible.

3.2.2. Gathering user data

One of the most important features of the *shepard Process Wizard* is guiding the user through the process. While the creation of the model can be done “offline” at the office, this part is intended being performed “online” in the lab or on the shop floor. The user is provided with a graphical user interface as it can be seen in fig. 3.
On the top of the window, the process step can be selected. The order of the process steps is determined by the model and done using a depth-first search algorithm, honoring the order of children defined using the predecessor/successor relation. If no such relation is defined, the order of children is undefined, therefore a random ordering is chosen. Using the `Next` and `Back` buttons, the user can navigate through the whole process. A direct selection of any process step is also possible.
Directly below the step selection, some information about the process step is displayed. This includes a description that can be specified by the process designer, as well as a direct link to the shepard frontend for the user, if a low-level access to the DataObject is wanted. Furthermore, times of the execution of this process step are displayed. Using the buttons `Start step` and `Stop step`, the current time can be saved either as start- or end-time for a process step. If a process needs to be interrupted, it can also be restarted later, leading to multiple execution times. The start and stop times are used for two purposes:

1) Information for the user, in particular to relate to the production process.
2) If time series data is available (e.g. by automatic recording of data of machines) the appropriate references will be created automatically. The references include the periods the step was active and therefore allow for an easy filtering of long running time series.

In the lower part of the window, the user can input all required data as it has been defined in the process type definition (cf. 3.1.1). Depending on the configured data type, the user is offered a field to enter text, numbers, a checkbox for Boolean values or an upload dialog to search for files for binary data. If the definition allows for it, the user can add additional lines to provide multiple items for the same field. Required fields are highlighted.
Sometimes – particularly in research applications – it is necessary to redo a whole production step, without starting a completely new process. Using the button `Redo step`, it is possible to create a new instance of the selected step (and its children) which can be filled with new data. To achieve this, a new DataObject, and recursively new DataObjects for all children, are created. The previous instances remain stored in shepard such that all data previously entered remains accessible.

4. SHEPARD PROCESS WIZARD WITHIN THE CONTEXT OF THE PROJECT NEUTRON

Development of *shepard Process Wizard* was carried out within the project NEUTRON. In the project, DLR aimed, among other objectives, to manufacture a demonstrator of a helicopter engine deck for a hybrid electric propulsion system.
The engine deck is an integral structural element of the upper fuselage. This platform-like structure provides the mounting interface for the engines via dedicated struts and brackets and establishes the primary load path to the main fuselage. It transfers operational loads from the engines into the helicopter’s primary airframe.
The demonstrator was realized in a new design made of carbon fiber reinforced plastic (CFRP). To this end, a semi-automated, quality assured manufacturing process was developed and implemented, encompassing the production of subcomponents (e.g. longerons and deck panel) and their joining into the final assembly. The process employed robot assisted dry fiber patch preforming combined with vacuum infusion using the VAP process. Final assembly was performed by riveting and a novel boltless joining method. Inline quality assurance was integrated into certain process steps, such as fiber angle measurement for textile cuttings. Figure 4 shows an overview of the implemented manufacturing process.
In order to map an industry-oriented process, the objective was to document all relevant information along the entire process chain, from the semi-finished product to the demonstrator, in order to ensure quality. Additionally, step-by-step guidance for manual operations was provided and automatic data logging for automated process steps enabled.
An example for a semi-automated process step was infiltration and curing of the part. Data like resin temperature, flow times, dielectric analysis of viscosity and similar was directly fed into the shepard database. Additionally, the graphical user interface of *shepard Process Wizard* as described in 3.2.2 provided the engineer with the possibility

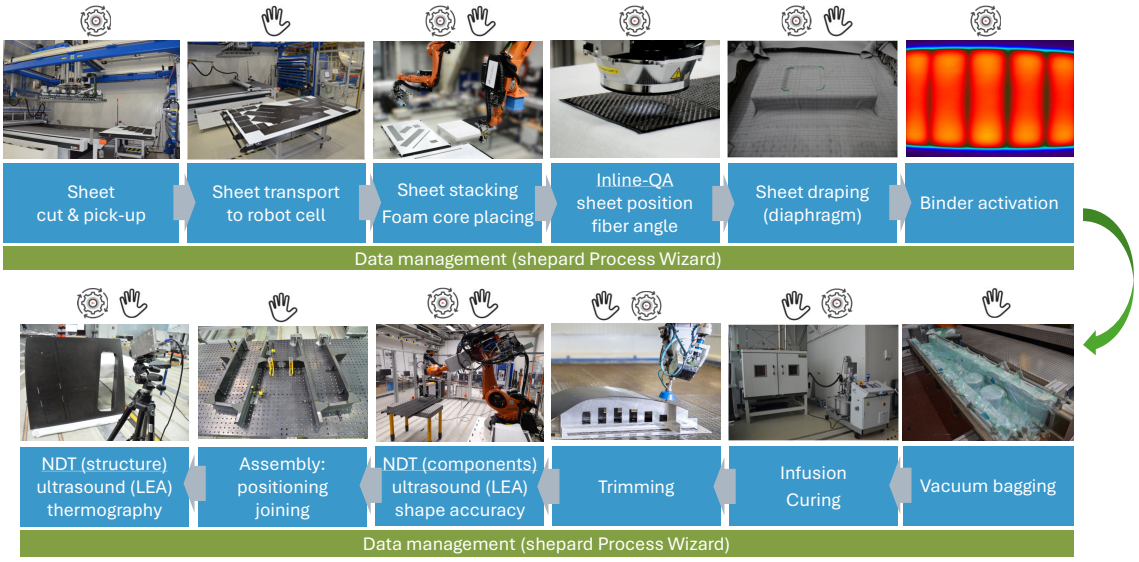

FIG 4. Manufacturing process of a CFRP helicopter engine deck implemented in Project NEUTRON

to log start and stop times, upload photo documentation of the setup, or add notes of unexpected occurrences.

5. CONCLUSION

The project NEUTRON established a complex manufacturing process with several heterogeneous steps which required a variety of data acquisition methods. The *shepard Process Wizard* allowed the creation of the data structure in shepard, including relations and references, in an intuitive and user-friendly way; thereby lowering the entry level for the application of the database without the need for extended training. Furthermore, the addition of manual steps enabled the use of the interface as an advanced possibility to document progress, record deviations or collect individual remarks. It is one more step towards fully automated data acquisition, without sacrificing the flexibility to amend the process with unforeseen additions. Short term communication with the test users yielded real time feedback from the manufacturing process; requested additional functionality or bug fixes could be implemented promptly. In conclusion, the close collaboration between developers and users offered an opportunity to test and improve during actual project operation, and help raise acceptance of the new tool.

Acknowledgment

Parts of this work were funded by the German Federal Ministry for Economic Affairs and Climate Action in the frame of the LuFo VI-2 project NEUTRON under the funding indicator 20M2114C.

Contact address:

michael.vistein@dlr.de

References

[1] Michael Vistein, Monika Mayer, Manuel Endraß, and Frederic Fischer. Single source of truth: Integrated process control and data acquisition system for the development of resistance welding of cfrp parts. In *20th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2023*, volume 1, pages 592–599. SciTePress, 2023. DOI: 10.5220/0012161500003543.

[2] Tobias Haase, Roland Glück, Patrick Kaufmann, and Mark Willmeroth. shepard - storage for heterogeneous product and research data, July 2021. Language: en. DOI: 10.5281/ZENODO.5091604.

[3] Florian Krebs, Mark Willmeroth, Tobias Haase, Patrick Kaufmann, Roland Glück, Dominik Deden, Lars Brandt, and Monika Mayer. Systematische Erfassung, Verwaltung und Nutzung von Daten aus Experimenten. In *Deutscher Luft- und Raumfahrtkongress 2021*. Deutsche Gesellschaft für Luft- und Raumfahrt, September 2021. DOI: 10.25967/550315.

[4] Michael Vistein, Patrick Kaufmann, and Tobias Haase. shepard Process Wizard, Sept. 2025. DOI: 10.5281/zenodo.17119446.