The requirements to process definition were analysed during meetings with domain experts for CFRP production. In one session, the method of "role playing" [6] was applied. Using this method, participating domain experts are assigned specific roles, having abilities they can offer and goals they have to achieve. A moderator triggers a conversation in which the participants are instructed to discuss how they can achieve their goals. In our case, there have been five different roles: a CFRP engineer, a robot arm, a gripping end-effector, a storage table and finally a tooling. The engineer had to instruct the robot arm and the gripping end-effector in a way that they can pick up a cut-piece from the table and drape it into the tooling. Using this method, we could observe the process work-flow and what kind of information is needed at which time by the different roles.

During this process, it became evident that by using a straight-forward approach beginning with picking up the cut-piece from the storage, some participants assumed information which was not readily available. Right at the beginning, there are infinite possibilities for fetching the cut-piece from the table. Once a *wrong* position has been chosen, it might not be possible to drape the cut-piece at all due to limitations of the tool geometry. However, detecting such a wrong position is only possible at the very last step. Solving these problems was possible by examining the whole preforming process starting with the last step and subsequently deducting prior steps.

III. Backward-oriented Task Programming

The results from the requirements analysis with CFRP domain experts suggested that it might be worth to put the focus on the *result* when planning complex manufacturing tasks and to derive necessary constraints for preceding tasks from the result's characteristics. This approach promises to be well suited for an interactive programming assistance that is geared towards the expertise and competences of manufacturing experts that have limited background in automation as such. Based on CAD models and further electronic data, many intermediate steps can be derived automatically and program code controlling the automation system can be finally generated. In this section, we introduce a novel approach for interactively programming complex manufacturing tasks using an offline programming environment. After a precise description of the general approach, we show the successful application to the CFRP domain.

A. Approach

Because manufacturing tasks are usually comprised of several sub-tasks, programming can be also split into several steps. For example, consider a task T which consists of n sequential sub-tasks T_1 to T_n. Every sub-task T_i can be programmed separately, if and only if the transitions from its predecessor T_{i-1} and to its successor T_{i+1} are valid. More formally, for every sub-task T_i the set of post-conditions $Post_i$ has to match the set of pre-conditions Pre_{i+1} of sub-task T_{i+1}, i. e. $Pre_{i+1} \subseteq Post_i$. Pre_0 is defined as the set of pre-conditions of task T, $Post_n$ as its set of post-conditions. A post-condition can, for example, be the final joint configuration of a robot after a sub-task. Accordingly, a pre-condition can be the initial joint configuration before a sub-task.

If it can be assured that all transitions are valid, every sub-task can be generated separately. A straightforward idea is to generate program code for sub-tasks in the same sequential order as they appear in the task. This *forward-oriented* task programming approach is simple to implement as the set of pre-conditions Pre_i for a sub-task T_i can be trivially derived from the set of post-conditions $Post_{i-1}$ for sub-task T_{i-1}. For every sub-task T_i, there is a solution space S_i that is comprised of all feasible programs that successfully accomplish the sub-task and thus establish the post condition $Post_i$. If the set of pre-conditions Pre_i is chosen arbitrarily, this can reduce the solution space S_i dramatically or even empty it. In particular, for problems where the solution space for subsequent sub-tasks is significantly smaller than the solution space of preceding sub-tasks, a forward-oriented task programming approach is not feasible.

Instead, we recommend for this kind of problems a *backward-oriented* task programming approach. Here, we start finding a solution or program for the sub-tasks with the smallest solution space. Without loss of generality, we assume that this is the last sub-task T_n. Having only post-conditions (i. e. the goal), an optimal solution for this sub-task can be found if it exists. Once it has been determined, it defines the set of pre-conditions Pre_n and the set of post-conditions $Post_{n-1}$. As the solution space S_{n-1} of sub-task T_{i-1} is usually greater than solution space S_n, the possibility of finding an appropriate solution is given. The technique can be applied until a solution for T_1 is found. In the field of artificial intelligence, this method is called *backward chaining* [7]. Our contribution is to successfully apply this method to an interactive and user-guided programming approach for complex manufacturing tasks. Its application and benefit is shown in the next section.

B. Application to the CFRP domain

In case of CFRP, engineers design composite products in special CAD software such as CATIA [8] or FiberSim [9]. These programs use draping simulations to calculate material properties and allow for exporting important manufacturing information (e. g. the plybook). As the plybook contains information about all cut-pieces, it serves as a recipe to create the initial task structure for manufacturing the complete CFRP part. Hence, the resulting structure conceptually consists of a sequential sequence of tasks, each responsible for the handling of a particular cut-piece. This contains the sub-tasks of *picking-up*, *transporting* and *draping* the respective cut-piece. Assuming all tasks having equivalent pre- and post-conditions, the programming of these tasks can happen independently in any order. However, the backward-oriented approach proposed above facilitates the programming of sub-tasks preforming a single cut-piece.

In order to specify the program (for handling a single cut-piece) in a conventional forward-oriented manner, the planar cut-piece has to be picked up from a storage onto a certain position of the gripper and moved to the tooling. In order to get the cut-piece transformed from its flat shape into its three-dimensional target shape, the engineer needs to set up a suitable deformation adjustment of the gripper with the cut-piece on it. Finally, the griper is positioned within the tooling and the cut-piece gets released. This process is performed repeatedly for every single cut-piece until the desired accuracy according

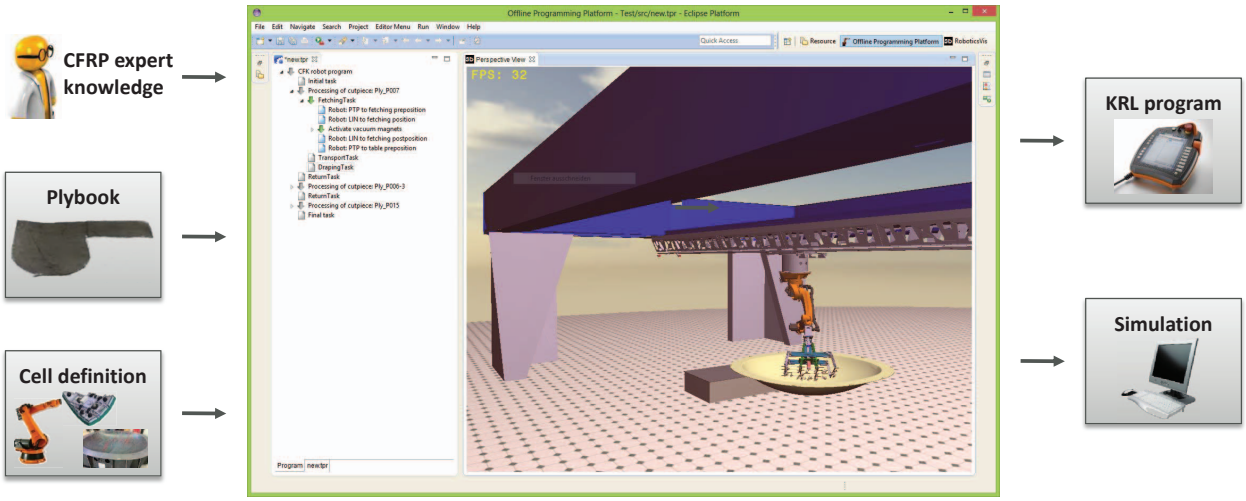

Figure 3. Suggested practise of robot programming supported by an offline programming platform

to the projected target contour is reached. We discovered that - depending on the pick-up position on the gripper - possibly no optimal solution or even no solution at all exists for precisely positioning the gripper on the target contour within the tooling. Using a forward-oriented approach, this is exactly the problem described in Sect. III-A. The position of the cut-piece on the gripper, which is an important precondition of the task, decreases the set of possible solutions. Therefore, we propose a backward-oriented approach which rather emphasizes the focus on the draping act and thus on accurate manufacturing results instead of focussing on the manufacturing process itself. However, using online programming, the reverse deformation of cut-pieces (from 3D to 2D) is not possible due to deviating and incomparable textile behaviour compared to the actual deformation from 2D to 3D. Thus, the offline programming platform is inevitably necessary for modelling such tasks in a backward-oriented manner.

For the backward-oriented approach, the programming starts with the cut-piece already positioned at the final correct position inside the tooling. The first and most important step is to define a specific gripper deformation and position within the tooling in order to perfectly place the cut-piece at its target position. Taking this as end-configuration, along with a freely specifiable start-configuration with the gripper close above the form, the deformation of the gripper from 2D to 3D and the draping movement can be retrieved from gripper-specific properties, which leads to the *draping* task. Currently, the draping set-up is interactively defined by the expert. Nevertheless, due to shape-matching or best-effort algorithms, an automated computing of this information without expert interaction seems feasible and remains as future work. Along with expert input where the cut-piece is initially placed on the supplier table, further data can be derived automatically by mathematical computations of the given information and three-dimensional models. Therewith, robot and gripper actions for picking-up the cut-piece from the specified position on the table (*pick-up* task) as well as robot movements for bringing it to the form while avoiding any collision (*transport* task), can be generated accordingly with regards to matching start- and end-configurations.

IV. Integration with Offline Programming

All requirements listed in Sect. II have been realized into a novel offline programming platform for developing CFRP manufacturing processes in a virtual environment. Besides the functionality of importing robot cell definitions and the plybook, a mechanism for automatically generating appropriate robot programs is desired as well. Furthermore, the platform must provide the possibility to graphically inspect developed tasks, for example by a simulation of the program within the three-dimensional robot cell. Based on the Eclipse IDE [10], an offline programming platform has been developed which integrates all requirements and the backward-oriented interactive approach in a research environment for developing CFRP manufacturing processes. The platform is shown in the centre of Fig. 3.

The backward-oriented concept (Sect. III) has been integrated into the platform to allow for autonomously generating task programs. To achieve this, the robot cell can be defined and the plybook can be imported (c. f. Fig. 3). The concept comprises several software modules, each of them having certain information as required input and being able to generate specific information. Some modules feature complex geometrical computations. Others require additional CFRP expert knowledge which can be obtained through a problem-specific user interaction. A logical linking of modules calculates sub-tasks for picking-up, transporting and draping cut-pieces and directly inserts the resulting sub-task into the initial task program structure.

The offline programming platform has an object-oriented model of hierarchical robot tasks as underlying programming language. A formal specification of the task structure was already given in Sect. III-A. Basically, tasks describe actions which are to be performed by actuators. This might be a linear movement of a robot arm, the deformation of a gripper, or the switching of field bus signals. By hierarchically nesting tasks, semantic relationships of tasks can be defined. For example, child tasks of a sequential parent task are executed strictly one after another. As known from common programming languages, each program instruction conventionally specifies a definite scope-depending post-condition which holds after the

particular instruction has been executed. In case of robot programming languages, each robot movement instruction knows the reproducible and unique robot pose, an end configuration, which will be reached after execution. Along end configurations, a task additionally specifies start configurations which represent the pre-condition that must hold before the task can be executed. The main purpose of start configurations is to support the individual steps of the proposed backward-oriented task generation (c. f. Sect. III-A). After the last step (i. e. the draping task for CFRP manufacturing) has been generated, its start configuration represents the end configuration of a previous task and thus is needed in order to generate this task (i. e. transforming and moving the gripper to the position where draping starts).

Both, the simulation within the visual robot cell and the code generation to proprietary robot code are available extensions of the platform. For simulating tasks and especially emulate time-critical movements and actions correctly, the *Robotics API* framework [11] is used. It provides full simulation support for robot arms and linear units, and it can easily be extended by further actuator support [12], as in our case by support for all three grippers. CFRP cut-pieces are also visually presented within the robot cell visualization and are simulated in a way that they initially lay on the table, can be picked-up and transported on a gripper. Regarding to performed gripper deformations, the cut-piece transforms from its 2D to 3D shape respectively. Besides simulating the whole task program, the framework also allows for simulating just parts out of the specified program. The virtual robot cell can immediately change to new device configurations correspondingly to the selected task. Starting at this configuration, the following movements and actions can be simulated successively.

In addition to simulation, the developed task program can be exported to run on real robot platforms. Task programs need to be transformed to language constructs of the target language. Currently, a code generator to the KUKA Robot Language (KRL) is provided by the offline programming platform. Extensions to the platform by further code-generators for other target-platforms are possible owing to the modular software design. When it comes to run the program in the real robot cell, the problem will arise that relative positions of objects within the virtual 3D model do not match their real positions accurately enough. To face this issue, all path-planning computations return object-relative positions and movements rather than absolute coordinates. When teaching the according real object positions, all computed results from the platform are then performed with adjusted corrections. The offline programming platform offers the possibility to play back positions modified manually so that even the simulated scenario in the visualization corresponds to the real cell. This does not effect the quality of produced code since it should be accurate anyway owing to relative movements. But an adapted visualization gives a better understanding of the real situation during development and enables a more realistic reachability analysis.

V. Evaluation

To evaluate the backward-oriented approach as well as the offline programming environment, a series of tests have been carried out at the ZLP facilities of the German Aerospace Center. The tests were run in two different robot cells, the so-called Multifunctional Cell (MFZ) [13] and the Technology Evaluation Cell (TEZ), using the three gripping end-effectors mentioned earlier. The set-up used for the test runs consisted of the particular end-effector mounted to a robot, a doubly curved geometry as tooling and a flat-shaped supplier table (cf. Fig. 4). All relevant objects of this set-up have been integrated into the offline programming platform, i. e. individual support for visualisation, simulation and code generation. To be able to verify both position and contour of the cut-piece put in place by the robot, a laser projection was installed that displayed the target contour onto the moulding tool.

Figure 4. Evaluating the Grid Gripper in the MFZ (top) and the TEZ (bottom)

Already the first run using the first cut-piece from the plybook was successful from a qualitative perspective. However, the cut-piece contour considerably varied from the target contour. The geometrical data used for the laser projection and the moulding tool was supposed to be inaccurate, so further effort was put into calibrating these objects using a laser tracker system. After calibration the process was repeated, leading to significantly better results. A detailed quantitative analysis is beyond the scope of this paper, but it is important to mention that the differences between cut-piece and target contour were hardly visible to the naked eye. That said, it is evident that the backward-oriented approach - without further optimization yet - works well enough to provide results comparable to the conventional, purely manual teaching approach.

In addition to the simplified expert-oriented programming, a significant advantage of our approach over teaching is the increase in speed. We have been able to achieve an average

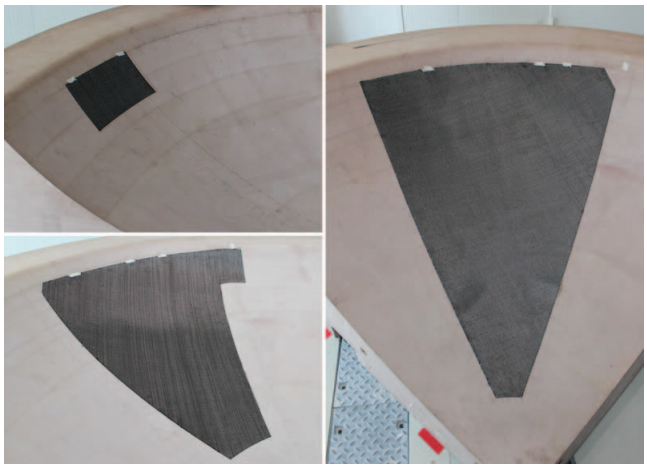

Figure 5. Three differently shaped cut-pieces used for evaluation

duration of around ten minutes for programming and handling of one cut-piece. Using the conventional teaching process, the duration varies from tens of minutes to several hours. Fig. 5 depicts three differently shaped cut-pieces, varying from $0.06m^2$ up to $1.5m^2$. By using the foam gripper, each cut-piece's preform process has been defined twice by teaching manually as well as using the proposed approach. For teaching manually, the code for activation of needed vacuum modules or setting up specific gripper deformations was created by hand. Additionally, the robot positions were defined. Since the cut-piece was invisible between gripper and form when searching for the draping position, it turned out to be a time-consuming repetitive process of approximating the cut-piece to the projected contour. Also, teaching could not make use of the advantages of backward-oriented programming owing to the non-reversibility of textile deformation mentioned in Sect. III-B. In contrast, the programming within the offline programming platform by using the proposed approach directly produced accurate results mathematically. The times it took in order to retrieve accurate results for both the conventional and the proposed approach are shown in Table I. Final test runs on the real robot cell, which are about 5 minutes, are not considered here.

Table I. TIMES NEEDED TO SPECIFY PREFORM PROCESS

cut-piece	conventional teaching	proposed approach
A	0:31 h	0:02 h
B	0:52 h	0:06 h
C	1:06 h	0:07 h

The measured times show an acceleration of the process by a factor of 8 up to 15. For production processes consisting of hundreds of different cut-pieces per part, the time needed for creating all robot applications shrinks from a month to a few days. This fundamental improvement makes in particular low batch size productions cost-effective and viable. Though, there is still further potential for speeding up the process, for example by automating the transfer of the robot program to the robot controller. Currently, the program has to be transferred manually by using some storage device.

VI. RELATED WORK

As the general idea of robot offline programming is nothing new, various software solutions exist for quite some time already. Almost every robot manufacturer offers his own ecosystem for offline programming (e. g. KUKA *WorkVisual* [14], or the *Robotics Suite* from Stäubli [15]). There are also manufacturer independent platforms, such as *Easy-rob* [16]. These powerful solutions are general purpose tools without domain specific process definition. Usually, the process definition has to be done on the abstraction level of basic robot movements (e.g. movement along linear, circular or spline paths). Some tools are geared towards particular domains (e.g. welding processes in case of RobotMotionCenter) or offer additional domain-specific software modules that are able to generate series of robot movements from domain specific input (e.g. the CAD data of a workpiece to be welded). However, those tools are far from providing a process-oriented guidance that is necessary for more complex manufacturing tasks that should be defined by domain experts without background in robot-based automation. To the authors' knowledge, an interactive backward-oriented planning support is not available in any of the existing tools. It might however be possible to extend some tools with software modules that provide such assistance.

For the domain of automated carbon fibre production, there exist offline programming solutions for Automated Tape Laying (ATL) and Automated Fiber Placement (AFP). Prominent examples are *Vericut* [17] and *MikroPlace* [18]. These tools support a domain specific process definition and provide certain guidance for domain experts [19]. However, ATL and AFP are significantly different production methods compared to the preform-based method this work deals with. Instead of working with large cut-pieces, these methods use narrow carbon fiber tape or "'tows"' which are fed from a magazine to the robot tool's head and then draped on the moulding tool. This process is in much respect similar to welding and has a lower complexity compared to preform-based processes, which comprise multiple heterogeneous and interdependent subtasks as well as sophisticated draping tools. Thus, robot programs for automated ATL and AFP draping processes can be calculated automatically based on geometric data.

VII. CONCLUSION

The proposed backward-oriented approach for offline programming of complex manufacturing tasks has been integrated into an offline programming platform in respect to several given requirements. In order to evaluate the approach in a real industrial scenario, the offline programming platform has been applied in a CFRP project as tool for specifying manufacturing processes. Due to emphasizing maximum automation and best supportive tooling by reliable mathematical computations and thus decreasing the dependency on often inaccurate human-made inputs, the evaluation showed an immense win in simplification of programming and thus time-efficiency for the task definition process.

We showed that the approach works well for the CFRP domain and further the use of the platform turned out to drastically minimize the effort for programming robot tasks to a less than a fraction compared to the current practise of industrial robot teaching.

Further work can be investigated to enrich the offline programming platform by an algorithm for specifying collision free movements or an extension which computes the best fit position of the gripper in the form for draping in order to minimize the need of interaction with experts. Also, the possibility to graphically edit robot cells within the platform would be a welcome nice-to-have.

ACKNOWLEDGMENT

We would like to thank Marian Köber, Dorothea Nieberl and Tobias Gerngross from the German Aerospace Center for their support. We also would like to thank Hella Seebach for giving advice and support during requirements analysis.

REFERENCES

[1] A. Strong, *Fundamentals of Composites Manufacturing, Second Edition: Materials, Methods and Applications*. Society of Manufacturing Engineers, 2008.

[2] S. Mazumdar, *Composites Manufacturing: Materials, Product, and Process Engineering*. Taylor & Francis, 2001.

[3] S. Lomov, *Non-Crimp Fabric Composites: Manufacturing, Properties and Applications*, ser. Woodhead Publishing Series in Composites Science and Engineering. Elsevier Science, 2011.

[4] T. Gerngross and D. Nieberl, "Automated manufacturing of large, three-dimensional CFRP parts from dry textiles," in *SAMPE EUROPE Technical Conference and Table-Top Exhibition*, Sep. 2014.

[5] DLR–ZLP. Project AZIMUT. [Online]. Available: http://www.dlr.de/bt/desktopdefault.aspx/tabid-8381/14345_read-36200/

[6] D. Leffingwell and D. Widrig, *Managing Software Requirements: A Unified Approach*. Addison-Wesley, 2000.

[7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[8] Dassault Systèmes. 3D CAD design software CATIA. [Online]. Available: http://www.3ds.com/products-services/catia/

[9] Siemens. Fibersim. [Online]. Available: http://www.plm.automation.siemens.com/de_de/products/fibersim/

[10] Eclipse. [Online]. Available: http://www.eclipse.org

[11] A. Angerer, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif, "Robotics API: Object-oriented software development for industrial robots," *J. of Softw. Eng. for Robotics*, vol. 4, no. 1, pp. 1–22, 2013.

[12] A. Hoffmann, A. Angerer, A. Schierl, M. Vistein, and W. Reif, "Managing extensibility and maintainability of industrial robotics software," in *Proc. 16th Intl. Conf. on Adv. Robotics, Uruguay*. IEEE, 2013.

[13] F. Krebs, L. Larsen, G. Braun, and W. Dudenhausen, "Design of a Multifunctional Cell for Aerospace CFRP Production," in *Advances in Sustainable and Competitive Manufacturing Systems*, ser. Lecture Notes in Mechanical Engineering, A. Azevedo, Ed. Springer International Publishing, 2013, pp. 515–524. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-00557-7_42

[14] KUKA Robotics. KUKA.WorkVisual. [Online]. Available: http://www.kuka-robotics.com/en/products/software/engineering_environment

[15] Stäubli. Robotics Suite. [Online]. Available: http://www.staubli.com/en/robotics/robot-software/pc-robot-programming-srs/

[16] EASY-ROB. EASY-ROB 3D Robot Simulation Tool. [Online]. Available: http://www.easy-rob.com

[17] CGTech. Vericut composite programming. [Online]. Available: http://www.cgtech.de/products/composite-applications/vcp/

[18] Mikrosam. Mikroplace. [Online]. Available: http://www.mikrosam.com/new/article/de/advanced-off-line-composite-programming-software/

[19] J. Sloan, "AFP/ATL Design-To-Manufacture: Bridging the Gap," *High-Performance Composites*, 2009.