 DOI: 10.3217/978-3-85125-524-9-06

RobWood - Smart Robotics for Wood Industry

Thomas Haspl[1], Claudio Capovilla[1], Alfred Rinnhofer[1], Victor J. Exposito Jimenez[1], Stefan Maier[2], Alexander Heinisch[2], Matthias Völkl[3], Manfred Zarnhofer[4], Robert A. Jöbstl[5], Erhard Pretterhofer[6], Bernhard Dieber[1] and Herwig Zeiner[1]

***Abstract*— Many branches of the manufacturing industry in general, and smaller wood processing companies in particular, are facing challenges related to producing ever smaller lot sizes under increasing time pressure. The *RobWood* project aims to increase the flexibility of such companies by providing a tool-chain to easily program robots for wood processing. In this paper we present an overview on our approach to robot programming by using models of the finished product.**

I. INTRODUCTION

Austrias wood processing industry accounts for 10 billion Euro, and ranks with a 3.9% trade balance surplus on second place just behind the tourism industry (4.2%). Each year, about 18,000 building construction permissions are issued, where prefabricated houses have a share of approx. 30-35%, with an upward tendency over the last years [2]. Ever higher demands regarding quality standards and individuality pose serious challenges to companies in the wood processing industry.

The goal of the *RobWood* project is to enable strong individualization of products at an equal or higher level of production efficiency through new technological approaches. The integration of robotics, sensor technology, and knowledge transfer with appropriate human-computer interfaces, applied in production, helps to optimize operating procedures in the wood industry. The use cases on which we work on in this project come specifically from the manufacturing of wooden prefabricated houses. Here, every house can be individualized but the parts are prefabricated in a factory instead of building them on site. In order to do this, many different steps have to be performed at each part like cutting, milling and clamping and the joining of different parts like steam brakes with the wooden elements.

Model-based programming is a powerful concept, which can lead to more natural interaction and easier programming of industry robots. Employees in smaller wood processing companies without in-depth knowledge regarding traditional robot programming will so be able to program robots themselves.

*The work reported in this article has been supported by the Austrian Research Promotion Agency under grant nr. 849896

[1] JOANNEUM RESEARCH
{firstname.lastname}@joanneum.at
[2] RIB-SAA
[3] ABB AG
[4] Zarnhofer Holzbau GmbH
[5] Haas Fertigbau Holzbauwerk GmbH & CoKG
[6] Holzcluster Steiermark GmbH

Research into intelligent technologies for accessing the data and knowledge created thereby has a strong leverage effect on its usage, already within single production enterprises and additionally across company boundaries.

The robot based production optimization pursued by the project has enormous potential regarding the creation of new jobs also in more rural areas, the efficient use of resources, and the transfer of insights to other sectors.

The rest of this paper is structured as follows: In section II we present related work, in section III we describe the challenges of automated wood processing, in sections IV and V we present the concept and tool chain of our solution and finally we conclude in section VI.

II. RELATED WORK

The trend towards computer based planning and processing methods has been finding its way into the wood manufacturing industry since a few years. In some sectors of the manufacturing industry, automated *CAD/CAM (computer aided design/manufacturing)* systems that generate machining data for use in lot size one manufacturing out of geometrical *CAD* data already exist - such as in the prefabricated concrete parts industry, and of course metal cutting on CNC machines as well as additive printing for prototype construction.

A. Model-based industrial Robot Programming

New research work is investigating new programming methods for making complex tasks easier to program for standard industrial robots [9]. Common approaches include offline programming methods with a complete 3D model [7]. The second common procedure called *teach-in*, or online programming, is very time consuming for complex task processing. Other approaches such as intuitive robot programming for *SMEs (small and medium-sized enterprises)* are described e.g. in [3], [13]. This approach is based on new types of e.g. gesture-based definition of poses, trajectories, and tasks. It is based on a visual programming concept that allows non-skilled programmer operators to create programs. For complex manipulation processes with a huge amount of *CAD* models, this approach does not significantly reduce the effort for the programming task.

B. Model-based Approaches

In short, *model-driven engineering (MDE)* [12], [14] is summarized as follows: model once, generate anywhere. This principle is particularly relevant when it comes to the building of robot applications. The modeling is done on different

abstraction levels and the (mostly) automated translation of models to machine readable codes increases efficiency for creation and maintenance of applications. Combined with the improved quality of implementation and reduced fault susceptibility, flexibility in production can be increased significantly. Another approach is the use of *domain specific languages (DSL)* [4]. These have been drawing attention especially in the area of service robotics during the last few years.

C. Complex Wood Manufacturing Processes for Robots

Typically a user of such a robot based environment has to perform different subtasks along a given manufacturing wood processing chain [6]. A technician may look at technical features, the customer indeed is mainly interested in how satisfactory they are solved. Automatically the user is scoring his satisfaction within the process execution, either in terms of time to perform, the process stability or comparing outcome quantity, efficiency and limitations. Complexity may be defined as the necessity to involve more than pure kinematic robotic control to perform a task, therefore going beyond the well-known operations.

III. PROBLEM STATEMENT

The main focus of innovative and new type of model based robot programming for wood manufacturing industries lies on the ease of use for non-experts in robotics coming from this specific domain. No knowledge about traditional off-line-programming or specifics of robot programming should be required. This requires the selection and implementation of an applicable method for creating the necessary data about manufacturing steps for the machining of solid wood elements with an articulated robot using different kinds of tools. As human labour is an integral part of manufacturing however, to make such a system available on the market today, interactive methods for the collaboration between human workers and a robotic system have to be examined and established.

Production steps where human interaction with the work piece is necessary should be kept at a minimum, where the machine operator can decide between using human labour where it might save time or material, while keeping the robot as fully engaged as possible. To ensure this, the commands for the robots need to be generated directly from the *CAD* plans, which have been enhanced with semantic information. Interfaces between *CAD* systems and robot installations need to use standardized formats as often as possible. In *CAD* there are various specific formats available, which need to be analysed, evaluated and may be adapted or enhanced for the intended use. The evaluation of existing open standards is also an important task in the *RobWood* project. However, future industrial use and uptake of the proposed technologies are subject to support by the *CAD* system providers. There are standardized interfaces in existence for use in *CNC (computerized numerical control)* production environments (DIN 66025/ISO 6983). These are mainly used for portal systems and toolsets and might need adaptation for usage with buckling arm robots.

Starting from a desired pose of an robots attached tool, one has to solve the so called inverse kinematics problem for a robot to obtain the corresponding robot axes configuration [5], [10]. This problem is difficult to solve mathematically, and typically has several solutions as shown in Figure 1 for an elbow up and elbow down configuration. A model-based approach, however, would offer this functionality for a broader spectrum of robot types and in particular, as accessible component earlier within the model-based software tool chain.

For the intended use case the number of produced unique pieces is one, although the reuse of parts of the models has to be considered. This also affects the specifications for creating enhanced user- and programming interfaces. Apart from specific variables such as technical interfaces, drivers, catalogues and configurations, the resulting user interface should strive to be as independent from the robot system as possible. System configuration should be kept to a minimum and is done with simple configuration procedures and minor manual settings. The process should not require an expert in robot programming or setup.

Special care is to be taken to ensure that the person resetting the system is not able to bypass security measures built into the system and that access to the robot for configuration is only available during idle times. Especially the manual steps required during a reset are to be built with simple visually enhanced instructions so that the wood manufacturing personnel can safely perform the necessary procedures without the help of experts in robotics.

The particular requirements of the wood manufacturing industry imply the necessity of a tool catalogue, which holds all required and possible tools as well as their corresponding procedure parameters such as speed of operation and logistics of operation. These catalogues can differ between system configurations, for example for the same procedure but requiring different tools. Bringing the various configurations into a form that is both readily comprehensible as well as comprehensive will be one of the challenges of the proposed project. To provide this system also for timber frame construction the various steps related to treatment, positioning and assembly need to be considered as part of the overall procedure, even though these tasks are not be fulfilled by the same robot system but with an assisting system, yet at the same time keeping the transparency for the user.

Special focus is granted to the usage of the envisioned systems in time sharing and collaborative environments, where different companies share one robot system or a specialised provider offers the robot system as a service. This increases the importance of using standardized interfaces to offer the simple exchange of treatment models and object data while at the same time ensuring *IPR (intellectual property)*. The vision for the final system is to integrate the whole class of production control systems and production planning systems. As various robot systems could be part of the same production not only the machine-to-human but also

the machine-to-machine communication has to be taken into account.

IV. THE ROBWOOD CONCEPT

The main task of the *RobWood* project is the development of several tools, which allow to combine each other in a very flexible and fast modifiable way.

A. A Model driven Approach

First of all, an overall controlling software, which is able to process and forward *CAD*-data, is needed. Within the *RobWood* project this software implementation has been named *Manufacturing Execution System (MES)*. A data sink from this *MES* has to be a robot cell controller, which takes care of all the robot related information such as the actual position or the life cycle status of particular tools. The cell controller should actually serve as abstraction layer between the *MES* and the robot platform. As many different robots can be applied, every robot would need a customized cell controller in order to meet the interface requirements of the *MES*. Additionally, a *Quality Inspection Unit (QIU)* has been developed. This is a vision based unit and is responsible for controlling and ensuring the lasting quality of the workpieces. In order to achieve a high grade of flexibility and reusability of data, a cloud service for exchanging *CAD*-data with other companies and users is implemented. This unit is called *Cloud Exchange Service (CES)*.

B. The Production Workflow

The combination and way of interaction of all parts involved in the *RobWood* project is visualized in figure 1, where all the gray shaded blocks indicate tools, which have been developed within the project.

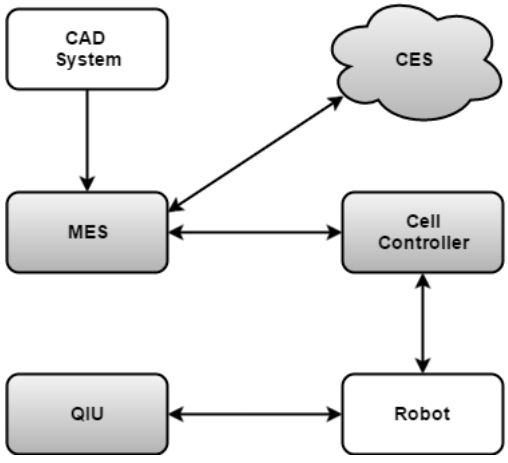

Fig. 1. Relations between the participating Units.

At the beginning, the designer designs, layouts and analyzes the house. During his/her work in the *CAD*-system this design is broken down into automatically producible elements. After handing over the elements to the *MES*, they are nested with other elements if possible and handed over to the production process. The production process is a quite broad term, starting from laser applications to plot the elements, augmented reality applications, printing plans or barcodes over to the actual manufacturing of the product. In this phase it is possible to interact with other plants or the cloud services to optimize production (e.g. by producing similar elements with specific characteristic in a plant dedicated to these). When the elements are taken over to the production process they are forwarded to certain cell controllers specified to interact with an unique robot or machine. These cell controllers are placed on site and give detailed insights in the actual work in progress of the robot. The main purpose of these controllers is to abstract the robots interface to the *MES* and give a more detailed insight for the user. The robot itself controls mechanical units needed to manufacture the product or to grant safety to the users.

C. Domain Specific Language

A *Domain Specific Language (DSL)* is a useful concept, which basically depicts a programming language that is created for a specific purpose. In other words, a *DSL* is a tool with limited focus, which we found to be an ideal opportunity for the *RobWood* project. Domain specific modeling is often used to describe concerns in robotics with concepts and notations to get closer to the respective problem domain and to raise the level of abstraction [8].

For the project we chose to implement a textual *DSL* instead of a graphical one as many frameworks are only available for the Java language runtime stack. A textual *DSL* also provides better integration to apply it in the future system which is based on the *.NET* language stack.

The most relevant issue in our selection process is the integration of the *DSL* in the *.NET* language stack. Another point that we have considered is the integration in a future platform. For these reasons we have decided to choose an internal *DSL* because we are able to implement our approach in a more affordable way. This kind of language fits very well for reactive systems, which are the systems that respond to external events, similar to robots. For all these aspects, the *F#* programming language was chosen to build our domain language.

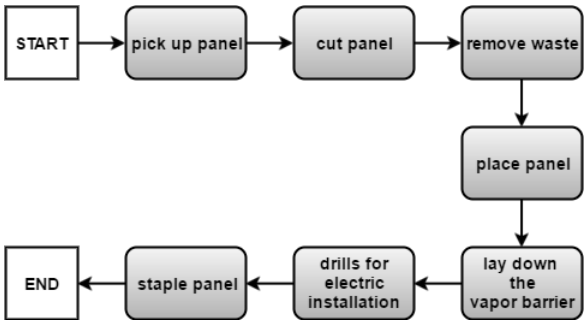

Fig. 2. Diagram of the Production.

In figure 2 the whole production process is visualized as a sequence of several tasks described by the *DSL*. These particular tasks can again be described as a composition of more fine-grained tasks. Such an example of the *pick up*

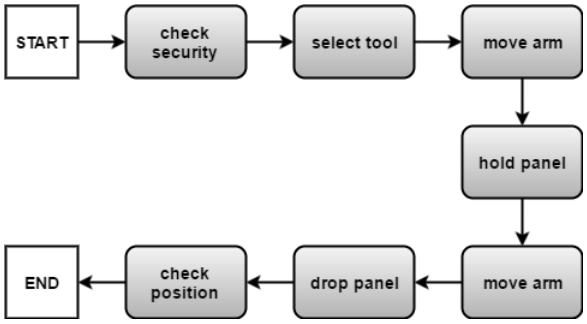

Fig. 3. Diagram of the *"pick up panel"* process as stated in figure 2.

panel task is shown in figure 3. Our *DSL* has three main commands to build each process. A ***process*** is used when a process starts. The name of the process is set as an attribute in the syntax. A ***task*** describes an automatized operation. The ***use*** command is used when the process is supposed to use a specific tool. Parameters can also be used to configure the behavior of the tool. So, the *pick up panel* process as in figure 3 can with the help of the *DSL* finally be written as follows.

```
process "pickup_panel"
task "check security"
use "arm" parametersv "5.5,6.0,8.0"
use "vacuum_griper" parameters "hold"
use "arm" parameters "3.5,3.0,8.0"
use "vacuum_griper" parameters "drop"
use "laser" parameters "0.0,0.0,0.0,1.5,1.5,1.5,1.5,0.0"
```

V. TOOL CHAIN

A. Manufacturing Execution System

The *Manufacturing Execution System (MES)* is responsible for preparing the *CAD* data for production and for performing and tracking the transformation of raw materials into finished goods.

As depicted in figure 4 the product is handed over by the *CAD* system. First, the *MES* checks, if the delivered data is correct regarding syntactic and semantic concerns such as closed contours or the considerations of the maximum producible dimensions. If these checks fail and an automated manufacturing process cannot be guaranteed, a meaningful report will be presented to the user, so that he can take further corrections.

After validating the *CAD* data, the system forwards the data to the *CAD* reader, which transforms it into an internal representation. Therefore, the beams of timber and panels of timber or gypsum are merged into elements according to their identifiers. Then, the surrounding contour of the elements is calculated. After that the beams of timber can be stored in the database. Since the beams are not part of the automatic manufacturing process, this information is only used for displaying them within the production units plan as well as to calculate the positions where the gypsum and timber panels have to be connected to the beams. The next step is to store the panels and the position (layer) of the steam brake in the database. Finally, all mounting parts for the particular processing steps are determined and written to the *CAD* file. This parts could be e.g. drillings for power outlets or heating systems.

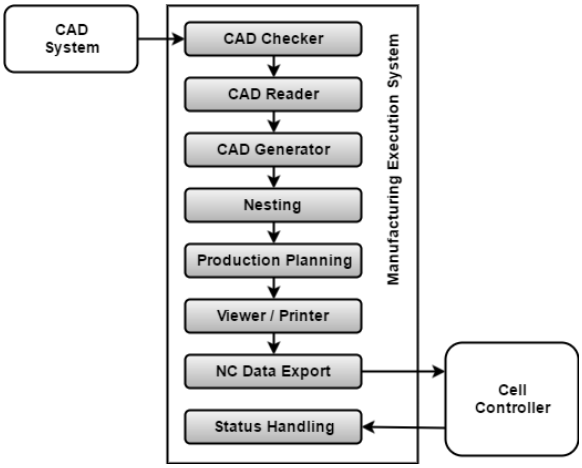

Fig. 4. Scheme of the Manufacturing Execution System.

The next unit within the *MES* is the *CAD* generator. In this step the positions for clamping are generated. Therefore, positions alongside the beams are calculated within a given distance. The dimensions of the steam brake is also calculated in this step. Since the steam brake is considered to be cut orthogonally only, and to be at least as broad as the element, this step can be reduced to calculating the length according to the elements contours surrounding rectangle plus some extra overhang.

The fourth part of the *MES* is called *Nesting* and enables the user to produce multiple elements to be processed on a single carrier. Elements are placed next to each other within certain constraints, which include dimensional restrictions, maximum number of mount parts per carrier or the minimum distance between elements. The latter is important to prevent an overlapping of the steam brake.

After the elements are nested on carriers the whole production process can be observed and planned in the production unit list. This list shows the current state of the production unit and it is also possible to view or print the plans of these production units, showing all the details of the elements contours according to their position on the carrier. Similar to this plans the carrier can be augmented by further information from a laser system.

The last unit of the *MES*, before the product is handed over to the production, is the *NC Data Export*. When the cell controller requests a new production unit, the *MES* prepares the next production unit in the production unit list to be produced.

Throughout the whole manufacturing process, the production unit can be tracked and its current status can be observed. Further, it is possible to analyze production, failures and throughput through a reporting service provided.

B. Cloud Exchange Service

The *Cloud Exchange Service (CES)* offers the opportunity for several companies to exchange different *CAD* modules,

the corresponding metadata and tool catalogues. The purpose of the *CES* is to lower the integration barrier of cloud services for *SMEs*. Its key functions are the management of *CAD* data. The *CAD*-files are of particular interest within *RobWood*, but also within non-spatial information such as processing sequences, which should be made more accessible and easily available through the cloud-exchange services. Cloud services may also be used to evaluate this data.

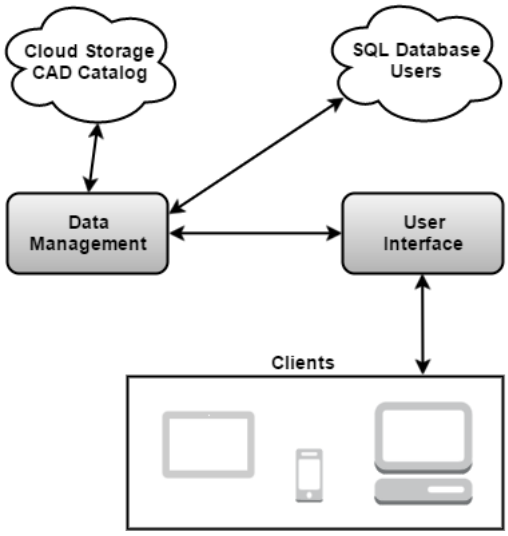

Fig. 5. Scheme of the Cloud Exchange Service Unit.

A general structure of the system is shown in figure 5. The system is developed on the basis of microservices architecture explained in [15], which provides more flexibility, resilience and scalability than a monolithic architecture. These microservices are small services that are in charge of small tasks. So, it is possible to build the whole system by the combination of them. The microservices are independent and connected through a *RESTful API* [11]. If needed, there can be added more microservices with few changes in the system architecture.

C. Cell Controller

The cell controllers main purpose is to request new data from the *MES*, optimize it for the specific robot platform and forward the prepared data to the robot. Besides that, it also serves as a human device interface, visualizing the status of the robot on a detailed level, giving the user more insight into the tasks performed by the robot. The information received from the *MES* contains all necessary data to automatically produce the elements on the carrier which is currently at the robots station. This includes all panels to be cut and placed, the steam brake and the drillings and clamping positions. Depending on the layer of the data blocks, their complexity and their position (e.g. path planning) a rearrangement of those can be done by the cell controller to increase cycle time and decrease waste. Each of these data blocks represent a single task to be performed at the robot.

Furthermore, the cell controller monitors the stock of raw materials and alerts the user if the cell runs out of timber or gypsum panels. Errors detected during production and the progress of the production unit are reported to the *MES* as well, depending on their severity and the kind of repair actions to be taken. Finally, after all tasks have been committed by the robot, the cell controller notifies the *MES* and requests the next production unit.

D. Robot

The robot is the last instance in the production line of the manufacturing process. It consists of a control unit able to receive new tasks from the cell controller. Each task consists of a single work package, e.g. contour and position of a panel. This package is handled by the robots control unit and split into single mechanical movements. The most important abilities of the robot are cutting, positioning, clamping and stapling the panels as well as placing the steam brake and drilling holes (e.g. for power outlets).

After the robot gets the task assigned to place a panel, it has to lift the panels out of the store. While doing so, it has to check if enough panels are on stock for future proceedings. If this is not the case it has to indicate the cell controller to notify the user. Since the refilling of the panel storage is not automated, the user has to fill these by hand. After lifting the panels, the robot places them on a dedicated work bench, the *carrier*, where the cutting is performed. The cutting is executed according to the contour information handed over by the cell controller. Depending on the material to be cut (e.g. timber, gypsum) the robot will change its tools automatically. Since the work bench is sloped down, the waste and dust emerging during cutting slips off the panel and has not to be removed explicitly. This allows that further layers can be positioned without an additional cleaning step. When all parts of a layer are positioned, the robot gets the instructions to nail the panels. Again, the tools are changed and staples or nails are driven through the panels into the beams to tighten them. Depending on the layering of the element the steam brake is applied after this step. To do so, the roll containing the steam brake is released and the robot pulls off the steam brake from the roll. After reaching the desired length, an orthogonal cut is made and the roll is locked again. After applying the steam brake, additional layers of panels can follow where the previous described steps have to be repeated. When all layers of the element are produced, the robot starts to drill holes through the panels and the steam brake before the production unit is released and handed over to the next production station of the plant.

E. Quality Inspection Unit

The *Quality Inspection Unit (QIU)* is used for supervising the clamping process required to combine different wooden elements. It consists of a number of components, which are shown in figure 6. Again, the units main parts are marked in gray.

A visual inspection sensor is mounted in or on the physical set-up in the production cell or even on the robot arm itself. This sensor data is processed by the *Sensor Control and*

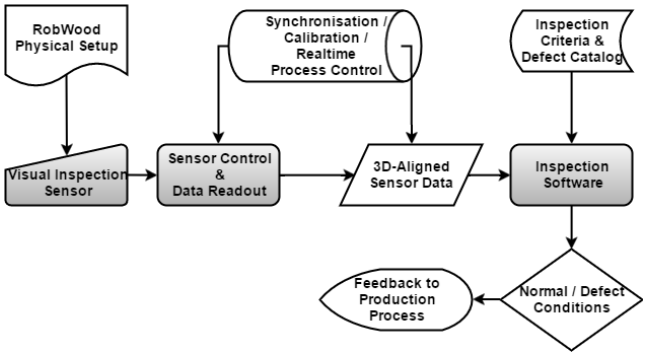

Fig. 6. Scheme of the Quality Inspection Unit

Data Readout unit that takes synchronization and calibration information from the control system to generate inspection sensor data aligned in 3D to the specimen to be produced.

The main part of the *QIU* is the *Inspection Software* that is fed with inspection criteria and quality thresholds and a defect catalog. The software decides upon defects and failures and reports those back to the production process. For fulfilling all these tasks the *QIU* provides a set of functionalities.

First of all, it captures the clamps once clamped, using a dedicated visual 2D/3D sensor. Afterwards, it generates a 3D representation, a so called *Digital Terrain Model (DTM)*, of the specimen surface. This *DTM* is then be analyzed with respect to segments that significantly exceed the ordinary surface plane of the specimen. Optionally a-priori information about the position of applied clamps can be used, to minimize the search space. In any case, the segments that indicate defects, are detected in this step. For such segments the responsible *MES* is beeing notified about the erroneous clamp, its position and optionally the amount and/or type of defect.

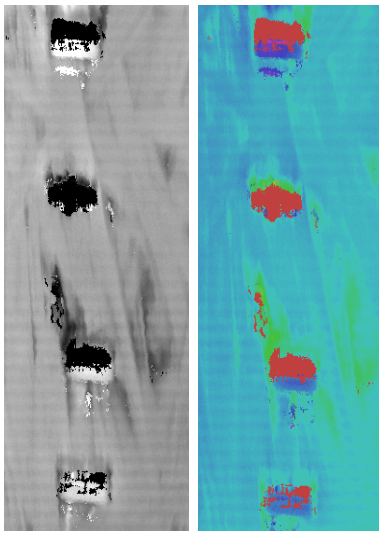

Fig. 7. Sample Images of not fully applied Clamps.

In figure 7 a sample of not fully applied clamps as detected by the *QIU* is shown. In the color coded image (right) the red segments represent undefined areas, which can be caused by an occlusion from a staple fully applied to generate a small ditch, or from an occlusion caused by a staple not fully applied. Dark blue areas are portions with higher elevation, hence not fully applied staples being detected as production errors . Light green means measured ditches. The measurement direction is from above.

VI. CONCLUSIONS

This paper contains an outline of the *RobWood* approach. We have described how we could design components which could be used for the company to program the robot with less effort. This is an important issue for workers without profound programming skills. A first, a series of tests of the particular components of the tool chain took place in a gradual manner at the *Holzinnovationszentrum* [1]. At the end, an integration test with all the components passed successfully. To sum up, we believe that our approach can be integrated in the production for the wood industry in the next three to five years.

REFERENCES

[1] Holzinnovationszentrum gmbh. [Online]. Available: http://www.hiz.at
[2] Proholz austria. [Online]. Available: http://www.proholz.at
[3] T. Dietz, U. Schneider, M. Barho, S. Oberer-Treitz, M. Drust, R. Hollmann, and M. Haegele, "Programming system for efficient use of industrial robots for deburring in sme environments," in *ROBOTIK 2012; 7th German Conference on Robotics*, May 2012, pp. 1–6.
[4] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.
[5] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 14–20, 1985.
[6] J. O. Huckaby and H. I. Christensen, "A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
[7] P. Neto, J. N. Pires, and A. P. Moreira, "Cad-based off-line robot programming," in *2010 IEEE Conference on Robotics, Automation and Mechatronics*, June 2010, pp. 516–521.
[8] A. Nordmann, N. Hochgeschwender, D. L. Wigand, and S. Wrede, "A survey on domain-specific modeling and languages in robotics," *Journal of Software Engineering in Robotics*, vol. 7, no. 1, 2016.
[9] Z. Pan, J. Polden, N. Larkin, S. V. Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87 – 94, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584511001001
[10] R. P. Paul and B. Shimano, "Kinematic control equations for simple manipulators," in *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, Jan 1978, pp. 1398–1406.
[11] L. Richardson and S. Ruby, *RESTful web services*. " O'Reilly Media, Inc.", 2008.
[12] C. Schlegel, T. Hassler, A. Lotz, and A. Steck, "Robotic software systems: From code-driven to model-driven designs," in *2009 International Conference on Advanced Robotics*, June 2009, pp. 1–8.
[13] M. Spangenberg and D. Henrich, "Towards an intuitive interface for instructing robots handling tasks based on verbalized physical effects," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, Aug 2014, pp. 79–84.
[14] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 461–466.
[15] H. Zeiner, M. Goller, V. J. Expósito Jiménez, F. Salmhofer, and W. Haas, "Secos: Web of things platform based on a microservices architecture and support of time-awareness," *e & i Elektrotechnik und Informationstechnik*, vol. 133, no. 3, pp. 158–162, 2016. [Online]. Available: http://dx.doi.org/10.1007/s00502-016-0404-z