

Institute of Software Technology

University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Bachelorarbeit

Evaluation and Integration of a Postprocessing Tool into a Container-based Cloud Platform

Nakharin Donsuypae

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Stefan Wagner
Supervisor:	Carsten Ellwein, M.Sc. Dipl.-Ing. Wolfgang Fechner
Commenced:	November 22, 2017
Completed:	May 22, 2018

Abstract

In times of Cloud Computing and emerging technologies, more and more manufacturing enterprises are in search of new ways to utilize said concepts and technologies to reduce time and money spent on the production of parts. As a result, Cloud Manufacturing as a new paradigm has been introduced and fundamentally changes the way how manufacturing enterprises do their business. With Cloud Manufacturing, networked manufacturing is made possible and companies are able to order production parts and manage virtualized resources directly on the manufacturing platform.

Rent'n'Produce, a research project of the Institute for Control Engineering of Machine Tools and Manufacturing Units, at the University of Stuttgart, focuses on the realization of a Cloud Manufacturing platform. Current functionalities include a flexible production assignment, a detailed order management and the possibility to schedule created resources.

To extend functionalities of Rent'n'Produce, this work will focus on the integration of a Postprocessing Tool to generate a Numerical Control programming language, which can be used to execute commands on a machine tool. Requirements are defined and different Postprocessing Tools are compared to find the most suitable solution for the integration. The most suitable Postprocessing Tool is then prototypically integrated in Rent'n'Produce to showcase the new implemented workflow.

Kurzfassung

In Zeiten von Cloud Computing und neuen Technologien, sind immer mehr Produzierende Unternehmen auf der Suche nach neuen Möglichkeiten um besagte Konzepte und Technologien umzusetzen, um Kosten sowie Zeit bei der Produktion von Werkstücken einzusparen. Infolgedessen, hat sich Cloud Manufacturing als ein neues Paradigma etabliert und verändert grundlegend die Vorgehensweise in Produzierende Unternehmen. Mit der Einführung von Cloud Manufacturing, wird eine vernetzte Produktion ermöglicht und Unternehmen können Werkstücke und virtualisierte Ressourcen auf der Plattform verwalten und beauftragen.

Das Forschungsprojekt, Rent'n'Produce, welches am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen an der Universität Stuttgart durchgeführt wird, arbeitet derzeit an der Realisierung einer Cloud-basierten Plattform für Fertigungsbeauftragungen. Aktuelle Funktionalitäten beinhalten eine hochflexible Fertigungsbeauftragung, ein detailliertes Auftragsmanagement und die Möglichkeit erstellte Ressourcen im Belegungsplan zu verwalten.

Um die Funktionalitäten von Rent'n'Produce zu erweitern, liegt der Fokus dieser Arbeit auf der Integration eines Postprocessing Tools für die Generierung einer Programmiersprache für die Numerischen Steuerung, welcher anschließend von einer Werkzeugmaschine verwendet werden kann. Mit der Definierung von Anforderungen und das Vergleichen von verschiedenen Postprocessing Tools wird das optimalste Tool für die Integration gefunden. Das passende Postprocessing Tool wird dann prototypisch in Rent'n'Produce integriert, um den neu implementierten Prozess zu veranschaulichen.

Contents

1	Introduction	17
1.1	Motivation	18
1.2	Goal	18
2	Fundamentals	21
2.1	Cloud Manufacturing	21
2.2	Rent 'n' Produce	23
2.3	Computer-aided Design and Computer-aided Manufacturing Process Chain .	25
2.3.1	Computer-aided Design	25
2.3.2	Computer-aided Manufacturing	26
2.3.3	Postprocessing Tool	26
2.3.4	Numerical Control programming language in International Organi- zation for Standardization format	26
2.3.5	Computerized Numerical Control	26
2.4	State of the Art	27
2.4.1	Spring Boot	27
2.4.2	React	27
2.4.3	Representational State Transfer	28
2.4.4	Container-based Virtualization	28
2.5	State of Research and Related Work	30
2.5.1	Manufacturing and the Cloud	30
2.5.2	Control Engineering in the Cloud	31
2.5.3	Cloud Manufacturing Platforms	31
2.6	Conclusion	33
3	Evaluation of Postprocessing Tools	35
3.1	Mandatory Requirements	35
3.2	Optional Requirements	36
3.3	Selected Tools	37
3.3.1	Overview	39
3.4	Evaluation	40
3.5	Manufacturing Parameters	42
3.6	Conclusion	44
4	Concept	45
4.1	Use Cases	45
4.2	Architecture	51

4.3	Integration	53
4.4	Discussion	57
4.5	Conclusion	60
5	Conclusion and Future Work	61
5.1	Conclusion	61
5.2	Future Work	61
	Bibliography	63

List of Figures

2.1	Architecture of a Cloud Manufacturing system, based on [Xu12]	22
2.2	Workflow of Rent'n'Produce, based on [ER17]	23
2.3	Simplified architecture of Rent'n'Produce, based on [ER17]	24
2.4	Computer-aided Design and Computer-aided-Manufacturing in the manufacturing process, based on [KRS15]	25
2.5	Virtualization architectures based on [XNR+13]	29
4.1	Target workflow of the integration	50
4.2	Target architecture of the integration	51
4.3	Resource View: Creation of a new resource	53
4.4	Tool View: Tool configuration window for tool parameters	54
4.5	Order View: Order Card window	55
4.6	Model View: Part configuration window for model parameters	56

List of Tables

3.1	Postprocessing Tools, based on set requirements	40
3.2	Defined Tool parameters	42
3.3	Defined Model parameters	43
4.1	Upload Computer-aided Design file	46
4.2	Configure Manufacturing Parameters	47
4.3	Convert Computer-aided Design file to Numerical Control programming language	48
4.4	Download Numerical Control programming language	49

List of Listings

2.1 Numerical Control programming language in International Organization for Standardization format	27
--	----

List of Abbreviations

- 2D** Two-dimensional. 25
- 3D** Three-dimensional. 25
- API** Application Programming Interface. 18
- CAD** Computer-aided Design. 18
- CAM** Computer-aided Manufacturing. 18
- CAX** Computer-aided-technologies. 24
- CNC** Computer Numerical Control. 25
- CPU** Central Processing Unit. 40
- DXF** Drawing Interchange File Format. 36
- G-Code** Numerical Control Programming Language in International Organization for Standardization format. 17
- HTTP** Hypertext Transfer Protocol. 28
- IGES** Initial Graphics Exchange Specification. 35
- ISO** International Organization for Standardization. 18
- ISW** Institute for Control Engineering of Machine Tools and Manufacturing Units. 22
- PDF** Portable Document Format. 24
- PLC** Programmable Logic Controller. 30
- RAM** Random-Access Memory. 58
- REST** Representational State Transfer. 18
- RnP** Rent'n'Produce. 17
- SOA** Service-oriented Architecture. 23
- SoC** Separation of Concern. 59
- STEP** Standard for the Exchange of Product model data. 35
- STL** Stereolithography. 35
- SVG** Scalable Vector Graphics. 36

List of Abbreviations

UI User Interface. 27

UML Unified Modeling Language. 23

YAML YAML Ain't Markup Language. 29

ZIP Archive file format created by PKWARE. 55

1 Introduction

With the introduction of Cloud Computing, companies have changed the way how computing resources are treated. Instead of purchasing and maintaining the whole IT infrastructure locally, companies can access resources through the internet [AFG+10]. Services are treated as utilities and are paid based on how much and how long the service is used [BYV08]. Due to these advantages and new emerging technologies, Cloud Manufacturing as a new manufacturing paradigm has been introduced [TZV+11].

Cloud Manufacturing is a concept which aims to transform the traditional manufacturing process into the cloud and enable users of the platform an intelligent and more comfortable way to manage manufacturing resources and production assignment [Xu12]. The realization of a shared resource pool, such as shared manufacturing resources and capabilities, and on-demand use of manufacturing services provide users the ability to manage and view resources at any time or place [TZV+11]. It is possible to follow the whole manufacturing process from the designing stage to the production and manufacturing companies can scale their production according to the clients' demand [TZV+11].

The vision to drive self-optimizing production systems forward and to manage them through a cloud platform is an important aspect of Cloud Manufacturing but has not been the focus of many research projects. With *Rent'n'Produce (RnP)*, and the currently available functionalities, companies of different industry sectors can use the cloud platform for flexible production assignment and detailed order management. To drive the vision forward, RnP aims to integrate features and functionalities to access machine tools directly from the cloud.

In this thesis, the creation of a concept to integrate a *Postprocessing Tool* into RnP is presented. This is done by creating requirements to evaluate the selected tools and determine their suitability for a possible integration. The most suitable tool is to be prototypically implemented as a *service* and encapsulated in a virtualized *Docker* container. The new implemented functionality should provide users of the platform a Numerical Control Programming Language in International Organization for Standardization format (G-Code), which can be used to control their machine tools.

1.1 Motivation

The Cloud Manufacturing platform and research project Rent'n'Produce provides customers and vendors in the field of manufacturing a variety of features. Features include a flexible production assignment, a detailed order management and the possibility to schedule created resources in the built-in resource plan. To showcase the whole manufacturing process managed through one cloud platform and to drive the vision of self-optimizing production systems forward, RnP aims to expand its features and add functionalities so that users of the platform can access and control machine tools directly from the cloud. With this objective, a machine readable code has to be generated, so that machine tools can be controlled and execute commands. Consequently, a Postprocessing Tool to generate machine specific code has to be integrated into the platform. This machine specific code is called G-Code and can be generated by a *Computer-aided Manufacturing (CAM)* software using an integrated post processor.

1.2 Goal

The goal of this thesis is to integrate a Postprocessing Tool into Rent'n'Produce. The new functionality aims to provide users of the platform the possibility to convert their *Computer-aided Design (CAD)* files into G-Code. To implement this feature, a CAM software with the option to generate G-Code has to be chosen and evaluated. Subgoals are defined as follows:

- Evaluation and selection of a suitable Postprocessing Tool which provides the feature to generate G-Code (International Organization for Standardization (ISO) 6983).
- Prototypical integration of the Postprocessing Tool in RnP and encapsulation in a Docker container with a Representational State Transfer (REST)-ful Application Programming Interface (API) to access the new implemented service.

Structure

This thesis is structured as follows:

Chapter 2 — Fundamentals In this chapter, the fundamentals of the thesis are explained. An introduction to Cloud Manufacturing, the research project Rent'n'Produce and the CAD-CAM process chain are provided. Furthermore, state of the art technologies, which are used for the integration and related work are presented.

Chapter 3 — Evaluation of Postprocessing Tools This chapter presents the gathered requirements for the selection of the Postprocessing Tool. Multiple tools are introduced and evaluated, and a tool for the prototypical integration is chosen.

Chapter 4 — Concept In this chapter, a concept and Use Cases of the new integrated functionality are introduced. An architecture design is presented and integration results are shown. In the last part of the chapter, a discussion of the results are provided.

Chapter 5 — Conclusion and Future Work The conclusion sums up the outcome of the thesis and a conclusion with the future outlook on this topic is provided.

2 Fundamentals

In this chapter, fundamental knowledge that is required to understand used terms and concepts are introduced.

An introduction to the term Cloud Manufacturing is presented in Section 2.1, and Section 2.2 provides information about the research project RnP. Section 2.3 covers the CAD-CAM process chain to fully understand the manufacturing process in the context of this thesis. Then, in Section 2.4 state of the art technologies and concepts, which are used for the integration, is covered. Finally, in Section 2.5, state of research and related work to this thesis is presented and functionalities of different Cloud Manufacturing platforms are compared to RnP.

2.1 Cloud Manufacturing

Cloud Manufacturing is a networked manufacturing paradigm which has emerged due to the introduction of new technologies, such as Cloud Computing, Virtualization and Service-oriented Technologies [TZV+11]. It refers to the distribution of manufacturing resources via the internet and allows users of a Cloud Manufacturing platform access to manufacturing services in a flexible way, paying only for the time the service is used [MBM+12]. Manufacturing resources are managed and encapsulated in a centralized way, and services can range from product design, manufacturing to testing [Xu12]. In a Cloud Manufacturing system there are three types of users, defined as *provider*, *consumer* and *operator* [WGRS13].

The provider is the one who owns and provides manufacturing resources on the platform. Providers can take the form of a person, an organization, an enterprise, or a third party [TCD+14]. Consumers are the people who subscribe to manufacturing services, that are offered on the platform and pay for the time the service is used [TZV+11]. The responsibility of an operator is to operate the platform and deliver services, which can be used by providers and consumers. Additionally, the operator is also responsible for the maintenance of technologies and services involved within the platform [TZV+11].

The architecture of a Cloud Manufacturing system illustrated in Figure 2.1 consists of the following layers: the *manufacturing resource layer*, *virtual service layer*, *global service layer* and the *application layer* [TMTR15].

In the manufacturing resource layer, manufacturing resources which are required for the manufacturing process, such as machine tools, are provided. Resources are either physical

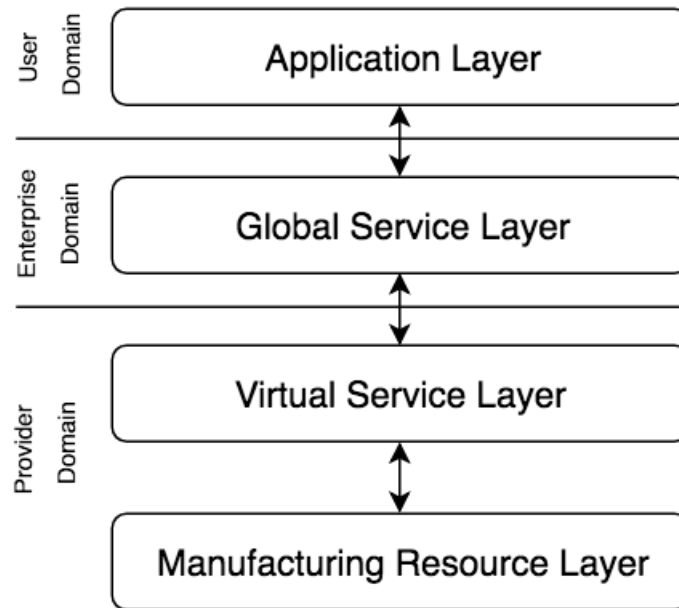


Figure 2.1: Architecture of a Cloud Manufacturing system, based on [Xu12]

or manufacturing capabilities. Physical resources include for instance, infrastructure, computers or servers [Xu12]. With manufacturing capabilities, companies offer capabilities for doing a special task and can include, product design capability, production capability, management capability, etc. [Xu12].

In the virtual service layer, manufacturing resources are identified, virtualized and then offered as cloud manufacturing services. These manufacturing services are then offered for users of the platform and are paid on demand [WGRS13].

The global service layer relies on cloud deployment technologies that are essential to run the manufacturing system. Cloud deployment technologies can be for instance, infrastructure as a service, where companies pay for remote infrastructure (storage, networking, etc.). This layer can operate in two operation modes, *complete service mode* and *partial service mode*. In the partial service mode, providers take some control of the processes in the system and the global service layer helps to manage and organize them. The complete service mode manages and organizes the whole Cloud Manufacturing activities itself [WGRS13].

Lastly, the application layer handles all interactions between the user and manufacturing resources. Users of the platform can construct and utilize manufacturing applications from virtualized manufacturing services [Xu12].

2.2 Rent 'n' Produce

Rent'n'Produce [ER17] is a Cloud Manufacturing platform and research project of the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), at the University of Stuttgart. Its goal is to offer companies in the field of manufacturing a cloud platform for flexible production assignment and act as a link between customers and vendors. So far, vendors are able to create manufacturing resources (e.g. machine tools), assign them to specific tasks and schedule them with a built-in resource plan. As seen in Figure 2.2, customers of the platform are also able to create order requests for the production of parts (bolts, nuts, etc.). These parts are then stored in a production pool, where vendors in the possession of a properly configured machine tool, can choose which parts they want to produce. If an agreement is concluded, a order request becomes a binding order and production of parts can start. Thus, location-independent production is made possible and can be entirely organized by the cloud platform.

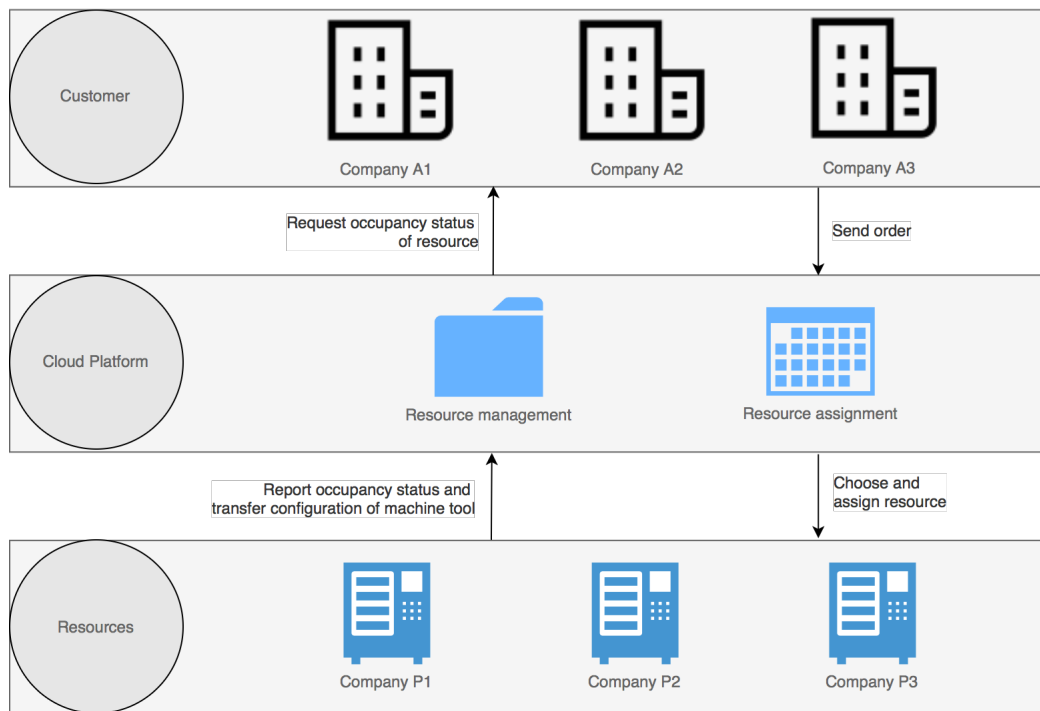


Figure 2.2: Workflow of Rent'n'Produce, based on [ER17]

The diagram shown in Figure 2.3, is a component diagram based on Unified Modeling Language (UML) 2.5.1¹ and showcases the Service-oriented Architecture (SOA) of RnP. As a result of this approach, services work independently from each other and have their own logic implementation. Each service is split up in a *persistence layer* and a *business logic layer*.

¹<https://www.omg.org/spec/UML/>

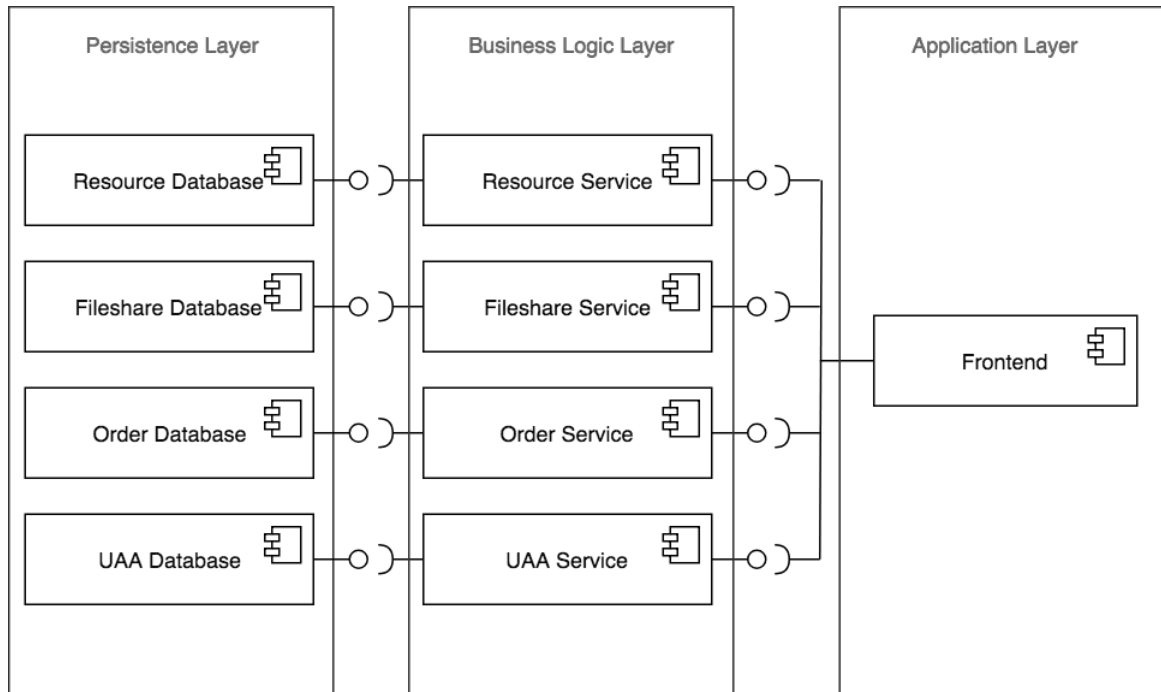


Figure 2.3: Simplified architecture of Rent'n'Produce, based on [ER17]

The persistence layer serves as a connection for the business logic layer to the database and is used for storing and accessing data from the database. The business logic layer contains all functionalities regarding implemented services and each service in this layer has logic specific implementation, which are used to do a specific task. The *Frontend* component is categorized in the application layer and contains the user interface and controls all url requests sent to the services. Its purpose is to handle all communications between the application layer and the business logic layer. To ensure platform independency and increase of maintainability, all services are encapsulated in Docker containers, using a *container-based Virtualization* approach [ER17].

All tasks regarding a resource is handled in the *Resource Service*. Tasks include for instance, creating a new machine tool, configuration of machine tools or managing the resource plan. Data created are then stored in a resource database. The *Fileshare Service* controls all documents and files (CAD file, Portable Document Format (PDF), sketch, etc.) regarding the order and order requests. All files are checked regarding their content type, to reduce vulnerability of the platform and then stored in the fileshare database. Order requests, orders and new offers are managed in the *Order Service*. Its purpose is to coordinate and organize all data regarding a order. Lastly, the *User Access and Authentication server Service* (UAA Service) manages all user roles (customer, vendor and administrator) and data of the user.

2.3 Computer-aided Design and Computer-aided Manufacturing Process Chain

This section introduces the CAD-CAM process chain for the production of parts in the manufacturing process. In the whole manufacturing process, different Computer-aided-technologies (CAx) are in use and the purpose of these technologies is to aid in the planning, simulation, analysis and manufacturing of a product [ES09]. For this thesis, CAD and CAM are the most relevant and therefore are presented in the following subsections.

In the context of machine tools, the CAD-CAM process chain describes the process starting from product design, programming, simulation to the production on the machine tool. As shown in Figure 2.4, the process for designing a product starts with the generation of a CAD file using a CAD software. In this stage, the user constructs a Two-dimensional (2D) or Three-dimensional (3D) model and defines model specific properties (material, tolerances, etc.). After the CAD file is created, the CAM software then uses the provided CAD file to create and configure a toolpath. To use the machine tool, a G-Code is generated by a post processor using the defined toolpath configuration of the CAM software. The generated G-Code is then used in the Computer Numerical Control (CNC) to run the pre-programmed sequences of commands for the machine tool. After the completion of these steps, a part is produced.

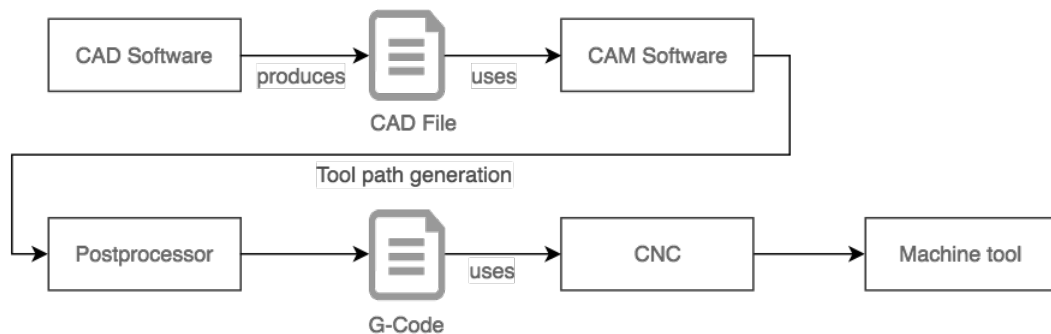


Figure 2.4: Computer-aided Design and Computer-aided-Manufacturing in the manufacturing process, based on [KRS15]

2.3.1 Computer-aided Design

Computer-aided Design is the use of a computer program to define, optimize or modify the geometry of a design and is used to create 2D and 3D models [KRS15]. CAD programs are used in different fields of manufacturing, ranging from the design of a simple bolt to jet engines. These programs allow users higher precision and improvement of the quality of design, due to the aid of optimization features in the CAD program [KRS15]. After the design of a model is completed, the output of a CAD program is a CAD file with a model that contains information, such as materials used, geometry dimensions and tolerances.

2.3.2 Computer-aided Manufacturing

The term Computer-aided Manufacturing refers to the creation and configuration of a toolpath [KRS15]. It uses a 2D or 3D model as a input and allows users to define used tools, geometry values of the model, coordinates of the bounding box and in some cases toolpath strategies. Additionally, an advanced CAM program is able to simulate the generated toolpath, thus collisions and errors in the milling process can be reduced. Furthermore, the generated toolpath can be converted to a G-Code using the integrated post processor and then be used in machine tools to execute pre-programmed tasks.

2.3.3 Postprocessing Tool

Post processing is the process to convert the generated toolpath of a CAM program into G-Code, the language of a CNC machine [KRS15]. The post processing functionality is often integrated in the CAM program, but some only offer the option to use functionalities of the program without the post processing part. Hence, the term Postprocessing Tool is used in this thesis as a definition for a CAM program with the post processing functionality.

2.3.4 Numerical Control programming language in International Organization for Standardization format

G-Code is the term for a programming language that is used to control automated machine tools [SSS07]. It is an industry standard and is often referred to as ISO 6983². Each line of code is executed sequentially and contains information for the machine tool on how to perform a specific task. Tasks include for instance, the position of the used tool, feed rate, spindle speed and rotation. In Listing 2.1, G-Code instructions are exemplified and a description for each task is provided. To showcase the task of each line, a description in the comments are presented. Comments are marked with semicolons and end by the start of the next line.

2.3.5 Computerized Numerical Control

Computer Numerical Control (CNC) is a method for automating machine tools [SSS07]. Using a CNC controller, a machine tool is able to read the G-Code and turn it into sequential commands to start production of a part. With the translated code, a machine tool is able to cut, shape or form parts made of different materials. Additionally, advanced machine tools have integrated tools for separate tasks which are used in different scenarios, such as cutting metal or wood.

²<https://www.iso.org/standard/34608.html>

Listing 2.1 Numerical Control programming language in International Organization for Standardization format

```
; G90: Use absolute position for XYZ axis
; G21: mm as a unit
G90 G21
; G01: Linear motion at feed rate
; X100: Go to position X=100
; Y75: Go to position Y=75
G01 X100 Y75
; G01: Linear motion at feed rate
; X200: Go to position X=200
; Y75: Go to position Y=75
G01 X200 Y75
; G00: Rapid linear motion
; Z:100 Go to position Z=100
G00 Z100
; M30: End of program
M30
```

2.4 State of the Art

This section covers the state of the art technologies and concepts, which have been used for the integration of the Postprocessing Tool. Each section provides a description of the technology or concept and advantages on why they were chosen.

2.4.1 Spring Boot

Services of RnP are implemented using the *Spring Boot*³ framework and Java⁴ as a programming language. Spring Boot is a solution for creating a stand-alone Spring application with minimal upfront configuration. It offers a range of features for developing backend services, some of which are embedded runtime support (Tomcat, Jetty or Undertow), production ready features (tracing, metrics) and simplified security features. Due to the minimal upfront configuration, the creation of services such as a simple Spring application with a REST-ful API and an integrated database support are facilitated.

2.4.2 React

For the Frontend development of RnP, *React*⁵ as a JavaScript library for building user interfaces has been used. React facilitates the design of reusable User Interface (UI) components, which means components are rendered and updated efficiently when data changes. Furthermore, React is component-based and encapsulation of multiple components that

³<https://projects.spring.io/spring-boot/>

⁴<https://www.oracle.com/java/>

⁵<http://reactjs.org>

manage their own state is made possible. Additionally, due to its ecosystem, a state manager such as *Redux*⁶ can be used for managing state components in the application and allows components to be centralized in a top-level component, called *Store*. Thus, all states are stored in the component *Store* and information can be requested when necessary.

2.4.3 Representational State Transfer

REST is an architectural style for *distributed hypermedia systems* based on the *Hypertext Transfer Protocol (HTTP)* [FT00]. *REST* is derived from different network-based architectural styles and is in contrast to many operation-based styles, an resource-based architectural style. Which means in *REST*, representations of a resource are transferred between a client and server. Applications that adapt *REST* as an architectural style are often called *REST-ful* applications and are characterized as stateless [RR08]. Consequently, the client data is not stored on the server between requests and *REST-ful* applications achieve higher scalability, because data is updated without affecting the whole system [MT10]. In *REST*, the client sends requests to the server to retrieve resources and the server sends back responses to the sent request. There are a set of operations which are supported using *HTTP* requests. Shown below are five types of *HTTP* requests to interact with resources in an *REST* architectural style.

- GET** – Retrieves a resource or a collection of resources.
- POST** – Creates a new resource.
- PUT** – Updates an existing resource.
- DELETE** – Deletes a resource.
- PATCH** – Modifies an existing resource.

Status codes from the server are returned after each request to alert the client about the success of each operation. The status code for a successful *HTTP* operation is for instance, 200 (OK) and a bad request is defined as 400 (Bad Request).

2.4.4 Container-based Virtualization

Container-based virtualization is a virtualization architecture that is considered a lightweight alternative to the *hypervisor-based virtualization* architecture [SPF+07]. With the *hypervisor-based virtualization*, each virtual machine has its own operating system that executes isolated from each other. As shown in Figure 2.5a, the execution of multiple different operating systems on a single host is possible and abstraction for the guest operating system is provided [XNR+13]. Container-based virtualization on the other hand, provides abstraction directly for the guest processes using containers on the application layer.

Containers are an abstraction of the application layer that allows applications to run with its dependencies in isolated processes. In this architecture, containers do not get their own

⁶<https://redux.js.org>

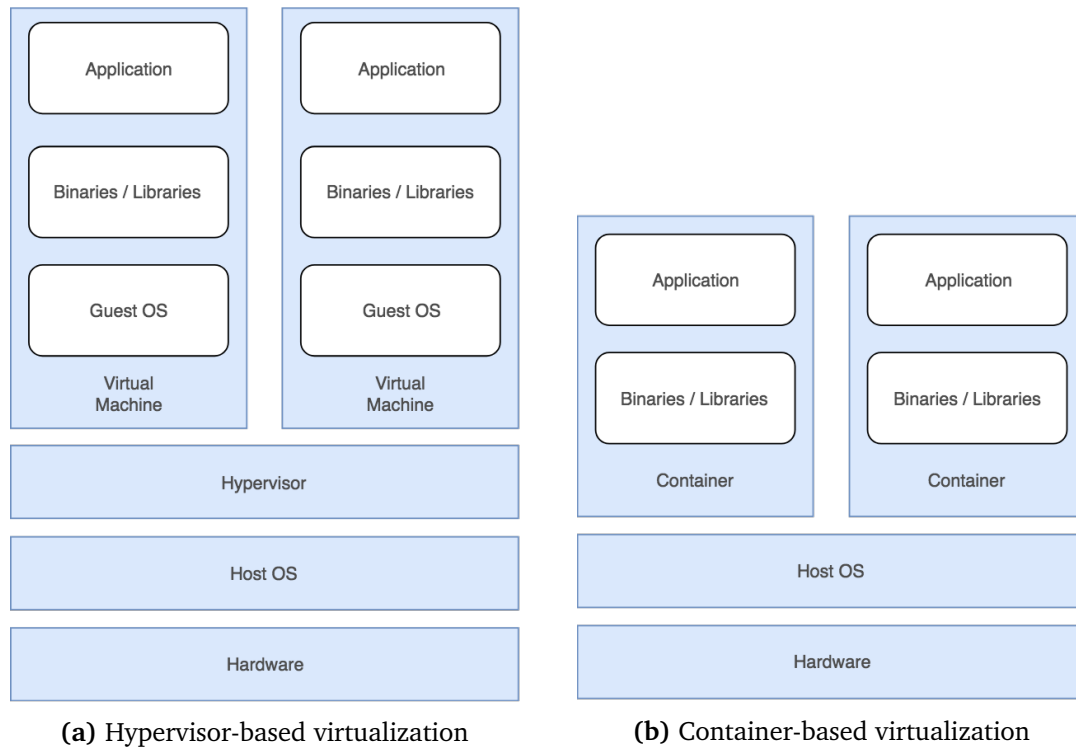


Figure 2.5: Virtualization architectures based on [XNR+13]

virtualized hardware, but rather use the hardware of the provided host system. As seen in Figure 2.5b, a container virtualizes the operating system and shares the host OS Kernel with the host and the container [XNR+13]. Because containers do not have to emulate hardware and boot a complete operating system, they can be started in less time and be more efficient compared to virtual machines [XNR+13].

Docker and Docker Compose

Docker⁷ is a containerization software to encapsulate software applications and facilitates the creation and deployment of applications by using containers. Docker containers are lightweight, platform independent and allows applications to run in isolated processes.

Docker Compose⁸ is a tool that is used to orchestrate multi-container Docker applications. With Docker Compose it is possible to define multi-container applications in a YAML Ain't Markup Language (YAML)⁹ file and get them to start running with a single command. According to the Docker documentation, Docker Compose works in all environments, such as production, staging, development and testing.

⁷<https://www.docker.com>

⁸<https://docs.docker.com/compose/>

⁹<http://yaml.org>

Because of the mentioned advantages, RnP uses Docker containers to encapsulate each service and Docker Compose to create and orchestrate all services.

2.5 State of Research and Related Work

This section introduces the current state of research in the field of manufacturing and control engineering in the cloud. Furthermore, related work in the context of Cloud Manufacturing platforms and the post processing functionality are presented.

2.5.1 Manufacturing and the Cloud

In the following, current manufacturing trends in the cloud, with the combination of other technologies and concepts, are introduced to give an insight into the current state of research and outlook of this topic and Cloud Manufacturing in general.

Researches are trying to find new ways to utilize new technologies to improve manufacturing structures and share manufacturing resources in an intelligent way through the cloud [RZW+17]. Adamson *et al.* [AWHM17] argues that to further drive *Smart Manufacturing* forward, different delivery models can be developed to support the integration of virtual, intangible and physical resources, such as CAD or CAM applications within a Cloud Manufacturing platform. Furthermore, with the aim to virtualize the whole manufacturing process, Xu *et al.* [Xu17] introduced a new era of manufacturing, where cyber-physical machine tools are solely managed by cloud-based solutions. An approach on how that can be done is proposed and includes the integration of the CAD-CAM process chain, as well as the access of a machine tool through the cloud platform.

Additionally, more researches aim for the realization of different CAx solutions and the integration of these solutions into a Cloud Manufacturing platform, to enable collective open innovation and rapid product development [AWHM17]. The results of this work can be applied to mentioned research interests, in which a cloud-based CAM solution is integrated into a Cloud Manufacturing platform.

Furthermore, decentralized production becomes more and more relevant. As a result, Yao *et al.* [YZZB17] describes a framework that allows a decentralized solution to further develop *Smart Manufacturing* and *Industry 4.0*. This framework integrates the physical, cyber and social systems as a whole and thus integrates humans, computers and machines, as well as human knowledge together.

With new emerging technologies, aspects of the traditional manufacturing process has transformed into the cloud and with the help of other concepts, Cloud Manufacturing can target a wider range of industries [TCD+14]. Current trends in Cloud Manufacturing have shown, that with the combination of other concepts and technologies, it can become intelligent manufacturing and has the potential to enable manufacturing companies to do business more cost effectively than before [WGRS13].

2.5.2 Control Engineering in the Cloud

Due to the advantages of Cloud Computing, such as the provision of virtualized resources and the possibility for on-demand usage, the field of control engineering is trying to move aspects of it into the cloud. Givehchi *et al.* [GITJ14] discussed an approach that implements the concept of *Programmable Logic Controller (PLC)* as a service. The result is a PLC solution based on the concepts of Cloud Computing. Although the solution showed slightly lower performance than a legacy PLC, it showcased that a cloud-based PLC solution is possible.

Another research project which has PLC as a focus is the project piCASSO¹⁰. piCASSO is a research project of the ISW, at the University of Stuttgart. Its goal is to enable flexible provision of control technology for cyber-physical systems in the production environment. With the use of Cloud Computing technologies and a service-oriented architecture, a cloud-based platform to control and navigate robots directly from the cloud has been developed. The results of the research project have showcased a different way to network and provision cyber-physical systems and showed that decentralized production controlled by a cloud platform can be realized.

2.5.3 Cloud Manufacturing Platforms

This subsection presents Cloud Manufacturing platforms related to the work of this thesis and showcases their functionalities in comparison to RnP and the implemented post processing feature. Additionally, a cloud-based CAD solution is presented to showcase different approaches of integrating the CAD-CAM process chain into the cloud. Furthermore, the following presented software solutions are not research projects and were developed by software companies with expertise in the area of manufacturing.

3DPrinterOS

3DPrinterOS¹¹ is a cloud printer management system to manage 3D printers, starting from design to the production. Since G-Code is also a file format used for 3D printers, a post processor is integrated into 3DPrinterOS to configure 3D designs and then convert them to generate G-Code. Thus, it follows the same objective as RnP in terms of accessing tools from a cloud platform. Features of 3DPrinterOS include a 3D printing management system, where users are able to connect their 3D printers to the cloud platform and access it on-demand. Additionally, stored 3D designs on the platform can be viewed and configured to the users' needs and production is started on-demand. Because of the integrated data tracking and analytics monitoring system, every step of the workflow is visualized and can be used for analyzing trends. Compared to RnP, the focus of 3DPrinterOS are 3D printers and therefore the generated G-Code is not suited for machine tools used for milling.

¹⁰<https://www.projekt-picasso.de>

¹¹<https://www.3dprinter0s.com>

Fabrikado

Fabrikado¹² is a manufacturing platform for ordering, producing and delivering parts. Its features include a price calculation system, which determines the price based on materials and product configuration. In comparison to RnP, where the focus lies on milling, Fabrikado targets milling and 3D printing processes. After uploading a 3D design, Fabrikado offers the functionality to do CAM specific configuration and adjustments of the product. Furthermore, Fabrikado uses a network of certified suppliers to produce parts. As a result of the matching system, customers are automatically assigned to a supplier with the cheapest production price. Although Fabrikado provides the functionality to adjust CAM configurations, it is unknown if a post processor is integrated or whether CAM values are sent to suppliers.

Fusion 360

Fusion 360¹³ is a cloud-based CAD and CAM software solution for the creation of a 2D or 3D model and configuration as well as generation of toolpaths. It has been developed by Autodesk¹⁴ and thus has a well-established reputation amongst the CAD and CAM community. With its current features, the complete CAD-CAM process chain can be followed through the cloud solution, starting from design of a product to the simulation of the milling process. However, to use these functionalities, one has to pay a subscription fee and pay for a selected period of time. Furthermore, since it is a sole CAD-CAM solution and lack the aspect of which RnP is built up on, such as ordering parts and scheduling resources, it is therefore not comparable in terms of the main objective.

Platform Comparison Conclusion

3DPrinterOS and Fabrikado have shown similarities with the target objective of this work, in offering users the possibility to configure their products directly on the cloud platform. Despite the available product configuration on Fabrikado, it is unknown if these parameters are sent to the supplier and thus questions arise if a post processor is integrated in the platform itself. Contrarily, 3DPrinterOS has an integrated post processor for converting uploaded 3D models to G-Code. But 3DPrinterOS mainly focuses on 3D printers and therefore differentiates between RnP in terms of the target group. The work of this thesis on the other hand, focuses on performing a Proof of Concept and demonstrate if similar workflow performed in 3DPrinterOS, by uploading CAD files and generate a configured G-Code, can be done in the context of CNC machine tools. In addition, Fusion 360 has been presented to showcase that the complete CAD-CAM process chain can be integrated fully into the cloud. As a result, the CAD-CAM process workflow starting from design to configuration of the post processor can be done using this solution. Since Fusion 360 is

¹²<https://www.fabrikado.com/de/>

¹³<https://www.autodesk.co.uk/products/fusion-360/overview>

¹⁴<https://www.autodesk.com>

a sole CAD-CAM cloud solution, it is not comparable to RnP, which has its focus on the integration of the whole manufacturing process. Nevertheless, all presented solutions have shown similarities to the work of this thesis and as a result showed that more companies are moving its on-premise solutions to the cloud.

2.6 Conclusion

This chapter introduced Cloud Manufacturing as a new networked manufacturing paradigm and presented the different types of users in these systems. Additionally, the four architectural layers have been presented and each layer was described in detail. Furthermore, RnP as a Cloud Manufacturing Platform and features including its SOA has been presented. Since the objective of this thesis is to integrate the CAD-CAM process chain into RnP, the CAD-CAM process chain and additional definitions were introduced. The state of research has shown the current research interests in the field Manufacturing and the Cloud and Control Engineering. Moreover, Cloud Manufacturing platforms and their functionalities including the post processing ability have been presented and compared to RnP.

3 Evaluation of Postprocessing Tools

In this chapter, requirements for the selection of Postprocessing Tools are collected and defined. Requirements are defined based on the suitability for RnP and the division between Mandatory in Section 3.1 and Optional in Section 3.2 is to distinguish the importance of each requirement. In Section 3.3, an overview of each tool is listed and aspects based on defined requirements are presented. Additionally, in Section 3.4 all tools are evaluated and the most promising candidate for the integration is selected. Section 3.5 covers the manufacturing parameters that are defined for the toolpath generation.

3.1 Mandatory Requirements

The following requirements are deemed as mandatory and are considered more important than optional requirements. Mandatory requirements are aspects that have to be considered due to their importance. For instance, since RnP is a cloud platform targeted for machine tools with the milling ability, aspects such as 3D file support and milling features cannot be ignored and need to be taken into consideration.

3D File Support

CAD files can be differentiated between 3D or 2D designs. 3D designs are generated by a CAD program and contain information on geometry values and properties, such as material, features or specific design details. Therefore, support for widely used CAD file formats are essential, so that a broad field of users can be addressed. A file format which is generated by most CAD programs and is standardized by the ISO is for instance, Standard for the Exchange of Product model data (STEP)¹. Other file formats which are popular amongst CAD programs and have to be taken into consideration are Stereolithography (STL) or Initial Graphics Exchange Specification (IGES).

Supported Interface

Interfaces are means to communicate between software components. Existing interfaces facilitate the integration and are therefore prioritized because no additional implementation

¹<https://www.iso.org/standard/63141.html>

has to be done to access components or features of the chosen solution. Furthermore, if additional implementation has to be done, critical features such as the post processing ability should not be altered, so that generated G-Code can still be used for machining.

License

Licenses, in particular software licenses, determine the right to use and the distribution of the software. Moreover, additional costs in the form of subscription fees are taken into consideration to estimate the budget necessary. Since RnP is based on free open source solutions, a Postprocessing Tool with no costs and a license to use, copy and change the source code is preferred.

Milling Features

RnP has its focus on machine tools with the ability to mill parts. Thus, milling features have to be compared and evaluated. Milling features are categorized in *basic*, *intermediate* and *advanced* and are used as a metric to define the quantity and quality of milling operations. Milling operations are defined as *basic* if a G-Code is created with minor configurations and the resulting G-Code can be used for a machine tool. *Intermediate* allows configuration of the G-Code on a more detailed level than basic and features, such as toolpath strategies, milling style, etc. are provided. Advanced milling operations are therefore features that exceed the post processing functionality. These features can be for instance, an integrated simulation tool or a collision detection system.

3.2 Optional Requirements

The following optional requirements are aspects that are useful but not necessarily needed. In comparison to mandatory requirements, the requirements defined here have lower priority and should be considered when a decision has not been made due to the high number of possible candidates.

Community Support

Active communities are important for the longevity and further development of the software solution. In addition, an active community contributing to an open source software is more likely to fix bugs once they have been reported. Consequently, communities have to be compared based on activity in case of a problem, community size, and if new releases are planned for the future or further development has been cancelled.

2D File Support

2D designs consist of curves and figures and are often used in machine tools for engraving. 2D file formats that are widely used and can be generated by most CAD programs are *Drawing Interchange File Format (DXF)* and *Scalable Vector Graphics (SVG)*. Due to the high amount of 2D file formats available, only the two mentioned formats are taken into consideration.

Additional Features

Additional features are functionalities that are offered in addition to the milling features. Furthermore, additional features are categorized the same as Milling Features in Section 3.1, starting from *basic*, *intermediate* to *advanced*. These features can be for instance, the visualization of a design or a file management system to organize created toolpaths.

Standalone Software

In the context of this thesis, a standalone software is defined as a software that does not require another software package or plugin in order to run. If one of the mentioned cases occur and for instance, a software package is needed, possible costs and licenses have to be re-examined to clarify if the selected tool in combination with the software package can be used.

3.3 Selected Tools

In this section, seven different Postprocessing Tools regarding the requirements in Section 3.1 and Section 3.2 are presented. As it is not feasible for the scope of this work to prototypically integrate each tool and test its functionalities, one tool is chosen and discussed in more detail in the following sections.

HeeksCNC

HeeksCNC [Hee] is a open source software that uses *wxWidgets*² for the graphical user interface (GUI), *OpenCASCADE*³ for solid modeling and several python modules for surface machining and pocketing operations. HeeksCNC is a plug-in for *HeeksCAD*⁴ and is therefore not a standalone software. STEP, IGES and DXF as file formats are supported and *advanced*

²<https://www.wxwidgets.org>

³<https://www.opencascade.com>

⁴<https://github.com/Heeks/heelscad>

3 Evaluation of Postprocessing Tools

post processing abilities are offered to configure the G-Code. No interfaces are supported and source code has to be extended if HeeksCNC is chosen for a integration. Development of the tool has stagnated since 2017 and no information on future development has been released.

PyCAM

PyCAM [PyC] is a open source software that specializes in toolpath generation for 3-axis CNC machining and has been developed using *Python*⁵. It supports STL as a 3D design and offers support for several 2D designs, including DXF and SVG files. *Advanced* milling features, such as collision detection and various toolpath strategies enable detailed configuration of the G-Code. Furthermore, all functionalities can be accessed through the *Command Line Interface* (CLI). Work on PyCAM has been paused between 2012 and 2017, but development has continued since the start of 2017, with an active community and contributors currently working on modernization of the code base and fixing bugs.

gCAD3D

gCAD3D [gCA] is a CAD and CAM open source software that provides an integrated post processor to convert STEP, IGES, DXF and SVG into G-Code. Furthermore, features such as a 3D model visualization tool are provided and *intermediate* configuration of the post processor can be done. The tool does not provide an interface where most features can be accessed and therefore source code has to be modified. gCAD3D is being developed by four people, and updates are released every few years.

Ace Converter

Ace Converter [Ace] is open source and provides the ability to convert DXF files to G-Code. Minor configurations of the post processor can be done using the CLI and no changes of the software have been made since 2015. According to the website, planned features are a toolpath visualization, file management system and better optimization algorithm. But it is unknown if these features are still being developed or have been cancelled.

FreeCAD

FreeCAD [Fre] is a CAD software with an integrated post processor to generate G-Code. FreeCAD has been developed with the use of open source libraries and the software itself can be used as a library by other programs. 3D formats that can be imported are STEP, IGES and STL, therefore all three 3D file formats mentioned in the *3D File Support* requirement

⁵[https://https://www.python.org](https://www.python.org)

are supported. In addition, a Python API is offered to access post processing features, but according to the website, this feature is still in the early stages of development. Nevertheless, milling operations are *intermediate* and created toolpath strategies can be simulated. In addition, FreeCAD has a active community, consisting of over 100 contributors, new releases every year and a forum that is regularly visited.

Dxf2gcode

Dxf2gcode [Dxf] specializes in the generation of G-Code based on 2D designs. As a result of this, only 2D designs, such as DXF, PDF and DWG are supported. Milling features are *basic* and the only parameters that can be set are boundaries, feed rate and tool diameter. Configuration of the post processor is saved in a configuration file and adjusted when required. Additionally, once the configuration file is set, the CLI can be used to trigger the post processing functionality. New features are frequently released and the internet forum is quite active, discussing bugs and future improvements.

FlatCAM

FlatCAM [Fla] is open source and focuses on printed circuit board milling. Supported file formats are Gerber and Excellon and does therefore not provide any support of the listed files names in mentioned requirements. Milling features are *intermediate* and include for instance, a visualization of the imported models. Additionally, all features can be accessed with the CLI and the internet forum is active with fast response time and frequent updates.

3.3.1 Overview

In Table 3.1, all tools are illustrated to showcase defined requirements and the comparison to each other. As a result of the analysis, each tool is free and open source and can be integrated with no additional costs. Additionally, only four of the seven compared tools support 3D designs, but 2D designs are supported by every tool, although they vary on file types. Furthermore, most tools can be accessed via an CLI, therefore source code does not have to be extended and features can be triggered by the CLI. *Intermediate* or even *advanced* milling features are provided by five out of seven tools and only one is not a standalone software. Due to active communities in four out of seven tools, development on future features are in progress and bug fixes happen on a regular basis.

Harvey Balls have been used to describe different categorizations defined in Milling Features and Additional Features. ● describes *advanced*, ◐ is *intermediate*, ○ is *basic* and N/A is Not Available. Furthermore, requirements were abbreviated and are defined as follows: 3D is 3D File Support, SuIn is Supported Interface, MiFe is Milling Features, CoSu is Community Support, AdFe is Additional Features and StSw is Standalone Software.

Table 3.1: Postprocessing Tools, based on set requirements

	3D	SuIn	License	MiFe	CoSu	2D	AdFe	StSw
HeeksCNC	STEP, IGES	N/A	GPLv3	●	No	DXF	◐	No
PyCAM	STL	Yes	GPLv3	●	Yes	DXF, SVG	◐	Yes
FreeCAD	STEP, IGES, STL	Yes	LGPL2	◐	Yes	DXF, SVG	◐	Yes
gCAD3D	STEP, IGES	N/A	GPLv3	◐	No	DXF, SVG	◐	Yes
Ace Converter	N/A	Yes	GPLv3	○	No	DXF	○	Yes
Dxf2gcode	N/A	Yes	GPLv3	○	Yes	DXF, PDF	◐	Yes
FlatCAM	N/A	Yes	MIT	◐	Yes	GBR	◐	Yes

3.4 Evaluation

In this section, the most promising tool for integration is selected. Individual benefits are highlighted and the decision process is explained in detail.

As a result of the evaluation, the tool which has been chosen and is considered the most suitable for integration is PyCAM. PyCAM with its wide ranging file support, allows RnP to target a broad field of users, making it possible for both 2D and 3D designs to be processed. Even though the only supported 3D file format is STL, PyCAM has the advantage of being a lightweight solution compared to other 3D supported tools, such as HeeksCNC and FreeCAD. Since HeeksCNC needs HeeksCAD to run, it is considered a heavier solution than PyCAM. CAD functionalities of HeeksCAD are not needed and the only features that are relevant for this thesis are the ones in HeeksCNC, which is only a small portion of the whole software. Same applies to FreeCAD, CAD features find no use in this current setup and to integrate a solution, which has not the main objective on post processing, is not optimal. Tools such as HeeksCNC and gCAD3D are considered not a suitable solution for integration, because interfaces to access main functionalities are not available. Hence, source code has to be extended, which makes it highly unfavorable over other solutions where features can be directly accessed with a CLI or an API.

Additionally, PyCAM provides *advanced* milling features, which go far beyond configuration of the post processor and offer more opportunities for detailed G-Code generation than tools, such as Ace Converter and Dxf2gcode. These features are e.g. a collision detection system, where all relevant triangles and directions are calculated to eliminate potential collisions, and a parallel processing feature, where calculations of toolpaths run in parallel based on available Central Processing Unit (CPU) cores. Furthermore, PyCAM offers toolpath strategies, where pre-defined configuration, called templates, are available and can be selected depending on the Use Case. As a result of this feature, users have the convenience of choosing a pre-defined template and facilitate the configuration process.

Even though development has stopped between 2012 and 2017, PyCAM announced future development in early 2017 and contributors are currently working on modernizing the code base. As a result of the announcement, the community has started to become more active and Issues in Github⁶ are processed in a short time. Moreover, a big community is capable of giving better support. In case of a problem it is very likely that someone in the community has already been working on it and solved it.

To summarize, all chosen solutions have unique features that make them distinct from one another. But previously mentioned benefits have shown that with its current functionalities, PyCAM is the most suitable tool for integration.

⁶<https://github.com>

3.5 Manufacturing Parameters

This section introduces manufacturing parameters, which are extracted from PyCAM for the adjustment of the post processor. For the prototypical integration, manufacturing parameters are divided between *tool parameters* and *model parameters*. As a result of this approach, milling tool configurations can be done in direct relation to the created resource and the adjustment of the model (processing area) is done after a binding order has been concluded. The following listed parameters will be used in the prototypical integration and define the input values a user can enter to configure the post processor.

Tool parameters in Table 3.2 define milling tool specific values. To each created resource, a milling tool can be created and associated with. These values have to be filled out, so that a tool specific G-Code is generated.

Table 3.2: Defined Tool parameters

Parameter name	Values	Description
Unit	Enum: [mm, inch]	Unit type that is set for all parameters
Tool Name	String toolName	Name of the tool
Tool Shape	Enum: [Cylindrical, Spherical, Toroidal]	Tool shape for the operation
Diameter	Integer diameter	Diameter of the tool
Torusdiameter	Integer torusdiameter	Torus diameter of the tool
Milling Style	Enum: [Conventional, Climb]	Milling style of the milling process
Safety Height	Integer safetyHeight	Height for re-positioning moves
Boundary Mode	Enum: [Inside, Along, Outside]	Specifies if the tool moves inside, around or along the defined processing area
Material Allowance	Integer materialAllowance	Minimum distance between the tool and object

As seen in Table 3.3, model parameters define details of the area which has to be milled. Boundaries of the processing area with its feed rate and spindle speed can be set. Additionally, a set of toolpath strategies on how the model is to be milled can be chosen.

Table 3.3: Defined Model parameters

Parameter name	Values	Description
Unit	Enum: [mm, inch]	Unit type that is set for all parameters
Boundary Type	Enum: [relative, absolute, custom] <u>Relative</u> : Margin of each face in percentage <u>Absolute</u> : Margin of each face in absolute units <u>Custom</u> : Absolute coordinates of the bounding box	Specifies the boundary type of the outer limits of the processing area
Upper Boundaries	Integer xUpper, Integer yUpper, Integer zUpper	Upper boundaries of the processing area using x, y and z as coordinates
Lower Boundaries	Integer xLower, Integer yLower, Integer zLower	Lower boundaries of the processing area using x, y and z as coordinates
Toolpath Strategy	Enum: [layer, surface, contour, engrave] <u>Layer</u> : Slice removal. Roughly cleaning of material around the model <u>Surface</u> : Detailed milling. Follows shape of the model by calculating lowest possible location of the tool at each position and mill around these locations <u>Contour</u> : Detailed milling. Detection of collisions in a fixed grid and connecting these points of collision to start the milling process around these points <u>Engrave</u> : Used for 2D models to do engraving	Defines available milling strategies on how the model is to be milled
Feed Rate	Integer feedRate	Movement velocity of the tool in units/minute
Spindle Speed	Integer spindleSpeed	Rotation speed of the tool in rotation/minute

3.6 Conclusion

In this chapter, requirements were defined as a selection criteria and further divided between mandatory and optional to distinguish their importance. Additionally, selected Postprocessing Tools were listed and important aspects including their features have been introduced. Each Postprocessing Tool has been evaluated and compared using defined requirements. As a result of the evaluation, PyCAM was chosen as the most suitable solution for integration. Additionally, manufacturing parameters for the adjustment of the post processor were extracted and described in detail.

4 Concept

This chapter describes the concept on how the Postprocessing Tool is integrated in Rent'n'Produce. In Section 4.1, Use Cases to showcase each functionality and associated actors are presented. Section 4.2 provides an architecture approach and a description to new implemented services. Furthermore, in Section 4.3, results of the integration are shown and a description of the new implemented services are provided. Section 4.4 discusses if all set goals have been reached or improvements can be made.

4.1 Use Cases

In this section, actors of Use Cases are defined and a description for each actor is provided. Additionally, a short summary in the end presents all listed Use Cases.

Customer: A customer is defined as a user of RnP who is able to create order requests, to which an offer can be generated. An offer can then either be accepted or declined.

Vendor: A vendor is defined as a user of RnP who is able to create offers for order requests. If an offer is accepted, it becomes a binding order and the vendor can start production.

System (RnP): The system, which is defined as RnP, is responsible for all sorts of communication and interaction between the user and system events. Additionally the system represents running services that are needed to perform a special task.

In the following, Use Cases are presented to showcase main functionalities and illustrate the main target workflow of the integration. Table 4.1 describes the process on how customers upload a CAD file to the system. To define manufacturing parameters and configure the post processor, Use Case in Table 4.2 presents necessary steps. After the configurations are done, Use Case in Table 4.3 converts a stored CAD file in the system to G-Code. Finally, Use Case in Table 4.4 showcases steps to download converted G-Code and extract them from the database.

Table 4.1: Upload Computer-aided Design file

Use Case Name	Upload CAD File
Description	Customer uploads a CAD file during the creation of a new order request. In addition, the uploaded CAD file is then stored in the Transformation Service Database.
Actors	Customer System
Precondition	<ul style="list-style-type: none"> • Customer is registered and logged in • Customer is in Order Request View
Postcondition	<ul style="list-style-type: none"> • Customer has uploaded a CAD file • System has saved uploaded CAD file in the Transformation Service Database
Main Course	<ol style="list-style-type: none"> 1. Customer has filled out all values regarding a order request 2. Customer clicks on the CAD-Upload button 3. Customer chooses a CAD file 4. Customer confirms his order request and a view with all set values is displayed 5. System checks file format 6. System saves all values and CAD file
Alternate Courses	<ol style="list-style-type: none"> 1a. Customer has not filled out all values and a new window is displayed telling the user to fill out missing values 6a. System receives a format that is not supported by the system and displays a new window telling the user to upload supported file formats

Table 4.2: Configure Manufacturing Parameters

Use Case Name	Configure Manufacturing Parameters
Description	Manufacturing parameters associated with a resource can be configured and is stored in the CAM-Configuration Database.
Actors	Vendor System
Precondition	<ul style="list-style-type: none"> • Vendor is registered and logged in • Vendor is in Resource View • Vendor is in Order View after a binding order has been concluded
Postcondition	<ul style="list-style-type: none"> • Manufacturing Parameters have been created and is associated with a resource • System has stored created manufacturing parameters in the CAM-Configuration Database • Systems starts the conversion of CAD files into G-Code
Main Course	<ol style="list-style-type: none"> 1. Vendor selects a resource 2. Vendor clicks on "Add new tool" button 3. Vendor fills out all relevant information regarding tool 4. Vendor approves his input with a button click 5. System saves all values regarding created tool 6. Vendor has a binding order and clicks on "configure model" button in the Order View 7. Vendor fills out all relevant information regarding model 8. Vendor approves his input with a button click 9. System saves all values regarding configured model 10. System starts the conversion to G-Code using sent manufacturing parameters
Alternate Courses	4a., 8a. Vendor has not filled out all values and a new window is displayed telling the user to fill out missing values

Table 4.3: Convert Computer-aided Design file to Numerical Control programming language

Use Case Name	Convert CAD File to G-Code
Description	The Postprocessing Tool converts a CAD file to G-Code using configured manufacturing parameters
Actors	System
Precondition	<ul style="list-style-type: none"> • CAM-Configuration Service is running • Transformation Service is running
Postcondition	<ul style="list-style-type: none"> • G-Code has been produced using sent manufacturing parameters • G-Code has been stored in the Transformation File Database
Main Course	<ol style="list-style-type: none"> 1. System sends manufacturing parameters extracted from CAM-Configuration Service to the Transformation Service 2. System extracts CAD file from the Transformation File Database 3. System parses sent manufacturing parameters and triggers the conversion of CAD file to G-Code 4. System stores generated G-Code in the Transformation File Database
Alternate Courses	<ol style="list-style-type: none"> 1a. System is not able to find manufacturing parameters in the database and displays a error message 2a. System is not able to find CAD file and displays a error message

Table 4.4: Download Numerical Control programming language

Use Case Name	Download G-Code
Description	Vendor has the option to download G-Code based on his manufacturing parameter configuration on resource and model.
Actors	Vendor System
Precondition	<ul style="list-style-type: none"> • Vendor is registered and logged in • Vendor is in Order View after a binding order has been concluded
Postcondition	<ul style="list-style-type: none"> • Vendor has successfully downloaded a set of G-Code files based on his configurations and associated milling tools to the resource
Main Course	<ol style="list-style-type: none"> 1. Vendor clicks on the "Download G-Code" button 2. System retrieves G-Code files from the database 3. System compresses all G-Codes in an archive file format and returns it to the user
Alternate Courses	<ol style="list-style-type: none"> 1a. Vendor receives a information that the G-Code is not generated yet 2a. System is not able to find generated G-Code and displays a error message

Use Case Diagram

The Use Case diagram in Figure 4.1, is shown to summarize all Use Cases in one illustration. The following diagram is based on UML 2.5.1 and represents a Use Case diagram. In the context of this thesis, the customer’s only responsibility is to upload a CAD file using a file format that is supported by the system. If a binding order has been concluded the vendor is then able to adjust the post processor based on his created resources (machine tool). Simultaneously, after all manufacturing parameters have been set, the conversion is automatically triggered, as shown in the diagram. If the conversion has been completed, the vendor is then able to download a set of G-Code files based on his sent manufacturing parameters associated to a resource.

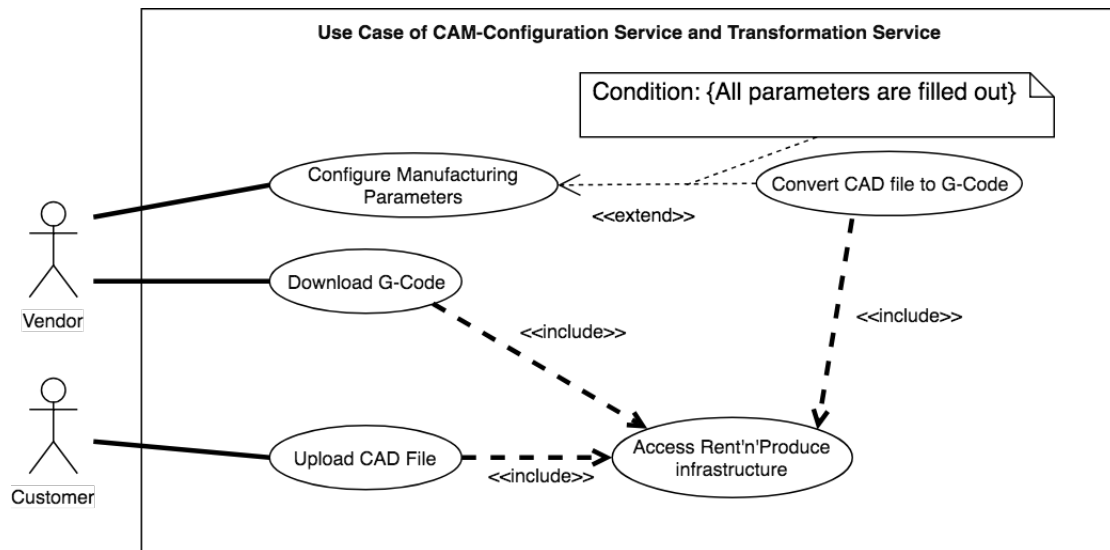


Figure 4.1: Target workflow of the integration

4.2 Architecture

This section focuses on the architecture approach of the integration. As seen in Figure 4.2, the *CAM-Configuration Service* and *Transformation Service* are separate services with each service having its own objective. Mentioned technologies and concepts introduced in Section 2.4, are used to implement these services. Therefore, each service is encapsulated in a Docker container and the means of communication between the Frontend and Backend services are via REST-ful APIs.

The main objective of the CAM-Configuration Service is to store all available manufacturing parameters in the database and handle API requests regarding tool and model configurations. Additionally, the CAM-Configuration Service serves as a central point of all communications regarding the creation of G-Code, which means parameters extracted from this service are sent to the Transformation Service to start the post processor.

The main task of the Transformation Service is the conversion of a CAD file into G-Code and the management of stored CAD files. CAD files, that are uploaded to RnP are stored in the *File Database*, which then can be converted into G-Code by a HTTP-POST request. Thus, the CAD file is retrieved from the database and sent manufacturing parameters from the CAM-Configuration service are parsed to start the conversion.

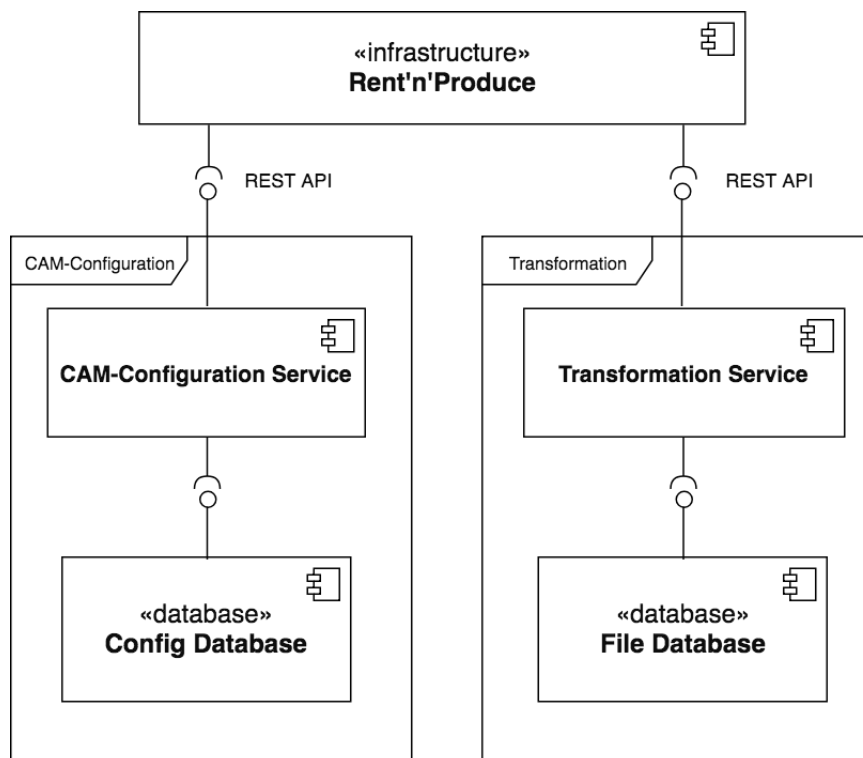


Figure 4.2: Target architecture of the integration

As a result of this approach, the configuration of manufacturing parameters and the conversion of G-Code are separated and services are less dependent on each other. With this architecture, multiple Transformation Services can be implemented and controlled by one CAM-Configuration Service e.g. since PyCAM only supports STL files, it is possible to add another Transformation Service, which offers the feature to convert STEP or IGES. These services can then all be managed by one CAM-Configuration Service. Furthermore, depending on which service is needed, a Transformation Service can be shut down to reduce the usage of computing resources.

Additionally, to package and run PyCAM system- and platform independent, a *Dockerfile* is defined to install necessary dependencies to run the software within a Docker Container. Since PyCAM is based on Python and Spring Boot is using Java as its programming language, Python dependencies and the Java JDK is installed when a Docker container is built from the Dockerfile. The resulting Spring Boot application contains the packaged PyCAM and is controlled via a REST-ful API, where functionalities of PyCAM are triggered via the CLI inside the Docker container. This approach allows a CAD file to be uploaded and then stored in the Transformation File Database. If manufacturing parameters are sent to the Transformation Service, then the associated CAD file is temporary extracted inside the Docker container and the manufacturing parameters are parsed to trigger the CLI. When the transformation to G-Code has completed, the resulting G-Code is then stored in the File Database, where it can be accessed using a HTTP-GET request.

4.3 Integration

This section showcases the result of the integration and all implemented features. A simple workflow consisting of creating a new resource and necessary steps for the conversion of a CAD file is shown. Features in the following figures, are shown with a red marked rectangle including a number as a reference on the far right corner.

As shown in Figure 4.3, to create resources, a vendor has to fill out information based on the machine tool he wants to create. If milling tools have already been created and assigned to this resource, a list is shown (1) with the names of already assigned milling tools.

In diesem Formular geben Sie alle nötigen Informationen an um eine Ressource anzulegen. Diese Ressourcen können nur von Ihnen eingesehen und bearbeitet werden. Bitte füllen Sie alle mit einem "*" markierten Felder aus.

Name der Ressource (*) ⓘ
Werkzeugmaschine A

Maschinentyp (*) ⓘ
Maschinentyp auswählen

Fehlertoleranz (*) ⓘ
10 µm

Mögliche Verarbeitung von (*) ⓘ
Kunststoff Metall

In diesem Bereich werden alle Werkzeuge aufgelistet, die für diese Ressource erstellt wurden.

Werkzeug A	(1)
Werkzeug B	

Werkzeug hinzufügen (2)

Abbrechen Vorschau

Figure 4.3: Resource View: Creation of a new resource

4 Concept

To assign a new milling tool to this specific resource, one can click on the button shown in (2) to open a modal window, shown in Figure 4.4, where tool parameters introduced in Table 3.2 can be set.

When all parameters are set, the new milling tool can be saved (3) and added to the list shown in (1). The vendor has now the opportunity to save his created resource and receive order requests based on the parameters of his resources.

In diesem Bereich geben Sie alle Informationen zum Werkzeug an, die für die Bearbeitung des Werkstücks verwendet werden können. Bitte füllen Sie alle mit einem "*" markierten Felder aus.

Werkzeugname (*)
Werkzeug C

Werkzeugform (*) Zylindrisch	Fräsart (*) Gleichlaufräsen	Durchmesser (*) 20	Torusdurchmesser 0
Maßeinheiten (*) mm	Werkzeugweg (*) Innen	Minimalabstand (*) 3	Sicherheitshöhe (*) 15

Abbrechen **Werkzeug hinzufügen** (3)


Figure 4.4: Tool View: Tool configuration window for tool parameters

In case of a match and an agreement between customer and vendor is concluded, the order request becomes a binding order. With this, the Order View, shown in Figure 4.5, is displayed and four different options are presented.

In (6), a new window to adjust model parameters, defined in Table 3.3, is opened. In this window, as seen in Figure 4.6, a 3D visualization of the CAD file is displayed and a list is presented to show all associated milling tools with this resource. In addition, the user has the option to display a toolpath strategy window (8), where all strategies are explained in detail. After all parameters have been filled out, the user can save his configuration (9) and return to the Order Card window. Simultaneously, the post processor starts converting given CAD file to G-Code, using set manufacturing parameters.

After the conversion has finished, the G-Code can be downloaded in (4). A G-Code is created for each milling tool to this resource. So, in case of four milling tools, four different G-Codes using the configuration of each milling tool are generated and compressed in one Archive file format created by PKWARE (ZIP)¹ file.

Kunde:	Musterfirma	
Abholbereit:	29.04.2018	
Maschinentyp:	Laserschneidmaschine (1 Achse)	
Materialien:	Kunststoff	
Stückzahl:	20	
Fehlertoleranz:	1 µm	
Gesamtpreis:	200€	
Ressource:	Werkzeugmaschine A	
Freitext:		



(4) G-Code herunterladen 	(5) Zur Produktion freigeben
(6) Werkstück konfigurieren 	(7) Produktion starten

Figure 4.5: Order View: Order Card window

To start the milling process on a machine tool, G-Code can be uploaded in (5). With (7), the uploaded G-Code is sent to the *Manufacturing Service*, where all G-Codes are added into a queue and transferred to the machine tool.

Note: The Manufacturing Service is being processed in parallel to this work and allows the transfer of G-Code to the machine tool and start production of a part using (7).

¹<https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT>

In diesem Bereich geben Sie alle Informationen und Maße des zu fräsenden Blocks an, welche für die Herstellung des Werkstücks benötigt werden.

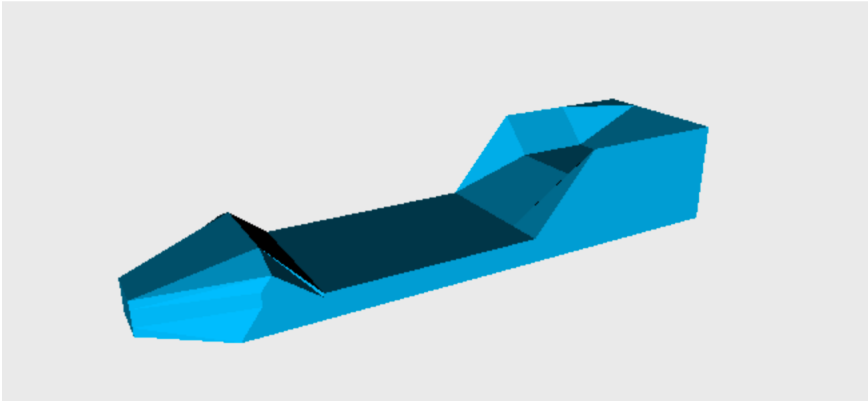
Maßeinheit (*) Bemaßungsart (*)

Untere Grenzen (X,Y,Z) (*) Obere Grenzen (X,Y,Z) (*)

Vorschubgeschwindigkeit in Maßeinheit/Minute (*) Spindeldrehzahl in Rotationen/Minute (*)

Bearbeitungsstrategie (*) Info zur Bearbeitungsstrategie:
 (8)

In diesem Bereich werden alle Werkzeuge aufgelistet, die für die Ressource: **Werkzeugmaschine A** erstellt wurden.



(9)

Figure 4.6: Model View: Part configuration window for model parameters

4.4 Discussion

This discussion focuses on the initial main objective of this work and compares the actual results of the implementation with previously set requirements and Use Cases. The results are discussed in detail and further improvements or limitations are presented.

Main Objective

To demonstrate the whole manufacturing process, defined in Section 2.3, RnP aims for the Shop-Floor integration, where machine tools can be accessed and directly controlled from the cloud platform. So far, a Office-Floor management, where users are able to create order requests, receive offers and conclude a binding order, is in place. To control machine tools, a machine readable code has to be created and generated. This step is essential and can be done by a CAM Software with an integrated post processor. Thus, this work had the focus on the integration of a Postprocessing Tool, so that machine language, G-Code, can be generated and configured on the cloud platform. With this addition, the generated G-Code can be sent to the Manufacturing Service, responsible for the machine tool and start the production.

Because RnP has been developed using mentioned technologies and concepts in Section 2.4, the integration had to be done following the same techniques. As a result, the new implemented services had to be implemented in Spring Boot and provide a REST-ful API to access functionalities. Furthermore, another requirement was to containerize services in Docker containers, so that services are encapsulated from each other and work independently. In addition, to demonstrate the new workflow, new views in the Frontend component had to be added. Following these requirements, an approach on how to compare, evaluate and choose the Postprocessing Tool had to be presented.

Approach

To achieve the objective mentioned in the previous section, this work divided the evaluation and integration in different chapters.

The evaluation, in Chapter 3, presented defined requirements to facilitate the selection of a Postprocessing Tool. These requirements were split in mandatory and optional to distinguish their importance. Reasons for that was to facilitate the selection process and provide a method to compare each tool. Additionally, since it is not feasible for the scope of this thesis to integrate each tool, one has to be chosen. Thus, setting up requirements was a necessary step for the selection of the most suitable solution.

In Chapter 4, Use Cases were defined to showcase the main target workflow of the integration. Moreover, an architecture has been presented, where CAM configurations and the transformation to G-Code are separated and thus leading to a separation of concerns

approach. Following the set architecture, the integration was carried out using technologies and concepts presented in Section 2.4.

Results

Use Cases presented in Section 4.1 serve as an indication if new implemented features have been fulfilled and the main objective as a whole has been achieved.

With the Use Case Upload CAD File in Table 4.1, the customer should have the opportunity to upload his CAD files in combination with his order requests. This Use Case was already implemented in the Frontend, therefore minor changes, such as a HTTP-POST method had to be added to direct the uploaded CAD file to the Transformation Service database. As a result, a REST-ful API to upload CAD files is provided and the goal of this Use Case has been achieved.

The Use Case Configure Manufacturing Parameters in Table 4.2 describes the process on how to adjust manufacturing parameters and create a milling tool associated with a resource. The result of the implemented Use Case can be seen in Figure 4.6 and Figure 4.4. Extracted parameters, defined in Section 3.5 can be used as an input and saved with a created resource. With this approach, configurations on the milling tool and on the model are separated and are automatically triggered if all parameters are set. Therefore, the conversion process starts in the background and users are informed when the generated G-Code is created. To summarize, this Use Case has been fulfilled and in addition a visualization view of the 3D model has been integrated. With this, users can view their 3D model and directly identify which CAD file has been initially uploaded. To further improve the features of this Use Case, a matching system can be considered, that checks all input values and recommends a resource on the platform for production.

The Use Case Convert CAD file to G-Code in Table 4.3 defines a technical Use Case involving the implemented services in RnP. If all manufacturing parameters are set and sent to the Transformation Service, the conversion of G-Code is triggered automatically. Thus, the CAD file is extracted from the File Database and the conversion is started by the CLI using sent manufacturing parameters. This process is done for each milling tool associated with the resource and finishes, when G-Code is generated and stored in the File Database. With this implementation approach, the post processing is dependent on available CPU cores and Random-Access Memory (RAM) of the Docker container and as a result varies in conversion speed depending on provided resources.

Use Case Download G-Code in Table 4.4 describes the process, in which a vendor is able to download G-Code based on his manufacturing parameters. As shown in Figure 4.5, a Download G-Code button has been provided and if triggered, the generated G-Codes are extracted from the File Database. Moreover, vendors can download multiple G-Codes simultaneously and receive a ZIP file containing all generated G-Codes associated with the resource of the order. Hence, this Use Case has been fulfilled and in addition offers the possibility to upload a G-Code and send it to the Manufacturing Service, to start the production on a machine tool. With this approach, a specific set of configured G-Codes can

be downloaded, but misses the individual aspect. For instance, if all parameters are set and one specific G-Code configuration has to be created, then all parameters have to be set anew. Therefore, a management system where all configurations are saved as a template and organized can be addressed.

Regarding the implementation, all services have been implemented in a way, where configuration of the post processor are managed in the CAM-Configuration Service and the post processing ability itself is managed in the Transformation Service. This architecture approach enables a Separation of Concern (SoC) design, leading to a structure in which added Transformation Services can be managed by one CAM-Configuration Service. For instance, if another Transformation Service, *TransformationSTEP Service*, is to be implemented to convert STEP files to G-Code, then this service can run in parallel with already implemented Transformation Service. These two services would receive manufacturing parameters from the CAM-Configuration Service and therefore a centralized way to manage all Transformation Services is made possible. In addition, all services run in Docker containers, following a virtualization approach described in Section 2.4.4 and thus enabling platform independency and encapsulation of each implemented service. To orchestrate and manage all Docker containers, Docker Compose has been added. Furthermore, the implementation follows REST as a resource-based architectural style, which means all sorts of communication between the new implemented services and the Frontend is done via REST-ful APIs. The integrated Postprocessing Tool, PyCAM, can be accessed through the CLI and thus, meeting all requirements which have been previously set.

Moreover, the evaluation in Section 3.4 has shown that PyCAM including its features provide more advantages than the integration of a sole post processor into RnP. Reasons for that are the offered functionalities of PyCAM including its additional features, in which a collision detection system is implemented to check for potential errors. In addition, a milling tool management system is integrated in PyCAM and thus offering unique features that are not provided in a post processor itself. Furthermore, the implementation of a post processor from the ground up is substantially harder than integrating an already established CAM solution in the open source community.

Regarding the state of research in Section 2.5 and the compared Cloud Manufacturing platforms, the final results have shown that the Proof of Concept of integrating a Postprocessing Tool into a Cloud Manufacturing platform was executed successfully. And thus providing an intermediate step in closing the manufacturing process defined in Section 2.3. With this functionality, machine tools are provided generated G-Code from the platform and therefore an essential intermediate step to the Shop-Floor integration has been added.

Limitations

Although all Use Cases and requirements have been met, there are still some improvements that can be made. PyCAM only supports STL as a 3D design and therefore limits the available designs a user can upload. Additionally, to process large CAD files, PyCAM needs a lot of time (approx. 5-15 min) depending on available CPU cores and RAM. Nevertheless,

one has to consider that PyCAM is a free and open source solution with a active community, which is currently working on improving offered functionalities of PyCAM. If the pace of progress is continuing, new supported file formats might be added in the future or even an increase of the post processing speed.

As Figure 4.5 has shown, a specific set of configured G-Codes can be downloaded, but lacks the individual aspect. Therefore, an additional management system to store and manage created templates in which individual configurations based on a specific number of milling tools can be addressed. With this addition, the user can organize and is able to do specific configuration on one or more milling tools.

Moreover, a matching system can be considered, so that set manufacturing parameters are directly matched with a created resource on the system. Thus, the vendor receives a recommendation on which machine tool is more suitable for the production of a part. This approach requires to alter the Resource Service and create an algorithm that matches created resources with set manufacturing parameters.

4.5 Conclusion

In this chapter, Use Cases to represent main functionalities and the target workflow have been presented. Additionally, a Use Case diagram with the target workflow of the customer and vendor has been illustrated. An architecture approach enabling the SoC design principle has been created and as a result of this approach, implemented services are not dependent on each other. The section Integration presented the integration results and the implemented workflow in RnP. In the end, the section Discussion presented a short summary of the main objective of this thesis and integration results have been discussed and compared to previously set Use Cases. To further improve the results, several approaches for future work have been described.

5 Conclusion and Future Work

This chapter provides a short summary of the work of this thesis and future work regarding improvements of the implementation is proposed.

5.1 Conclusion

In this work, a concept for the selection and integration of a Postprocessing Tool into RnP has been proposed. The integrated solution should provide the ability to convert CAD files to machine readable G-Code.

To accomplish this objective, mandatory and optional requirements have been presented as a selection criteria to choose the most suitable Postprocessing Tool. Additionally, a list of Postprocessing Tools was presented and evaluated based on defined requirements in previous chapters. As a result of the implementation, PyCAM as the most suitable solution has been identified. Manufacturing parameters were extracted from the chosen Postprocessing Tool and distinguished between model and tool parameters.

In the concept, Use Cases to illustrate the target workflow were shown and an architecture approach to integrate PyCAM has been described. Additionally, due to the provided advantages of PyCAM and its CAM functionalities as well as integrated post processor, the decision to integrate a sole post processor has been dwarfed by PyCAM and its additional CAM abilities. Furthermore, an architecture concept has been presented, in which a SoC approach is implemented to separate the CAM configuration and management together with the post processing ability. Consequently, services work independent from each other and can be managed by one service. Using the proposed architecture approach, PyCAM has been integrated into RnP and results of the implementation were presented. In addition, the results were then discussed and future improvement have been proposed.

5.2 Future Work

The current result of the integration have shown, that only a limited selection of file formats are supported and therefore limits the import ability of the user. Additionally, conversion of a large CAD file takes a substantial amount of time, leading to a long response time for the user.

5 Conclusion and Future Work

To improve mentioned deficits, additional Transformation Services can be implemented with each service using post processing capabilities specializing on a specific file format. Depending on the Use Case, additional Transformation Services for STEP or IGES could be implemented. With this addition, a wider field of users can be addressed and management of manufacturing parameters can be done using a single CAM-Configuration Service. Furthermore, with this approach, the workload on the implemented Transformation services is distributed and therefore reduces the time needed for a conversion.

Additionally, to facilitate the decision process of the user, a matching system that recommends a specific resource to given manufacturing parameters can be addressed. An algorithm where input manufacturing parameters are directly analyzed and runs through all created resources to check for a potential match and return the most suitable machine tool for production, would not only facilitate the workflow of production but also would create a better user experience.

Bibliography

- [Ace] AceConverter. *DAK Engineering - Ace Converter*. <http://www.dakeng.com/ace.htm>. (Accessed on 04/13/2018) (cit. on p. 38).
- [AFG+10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. “A view of cloud computing.” In: *Communications of the ACM* 53.4 (2010), pp. 50–58 (cit. on p. 17).
- [AWHM17] G. Adamson, L. Wang, M. Holm, P. Moore. “Cloud manufacturing—a critical review of recent development and future trends.” In: *International Journal of Computer Integrated Manufacturing* 30.4-5 (2017), pp. 347–380 (cit. on p. 30).
- [BYV08] R. Buyya, C. S. Yeo, S. Venugopal. “Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities.” In: *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*. Ieee. 2008, pp. 5–13 (cit. on p. 17).
- [Dxf] Dxf2gcode. *dxf2gcode download | SourceForge.net*. <https://sourceforge.net/projects/dxf2gcode/>. (Accessed on 04/13/2018) (cit. on p. 39).
- [ER17] C. Ellwein, O. Riedel. “Rent’n’Produce: A Social Media Platform for Cloud Manufacturing.” In: *Smart Materials and Nanotechnology in Engineering*. 4th International Conference on Smart Materials and Nanotechnology in Engineering (SMNE 2017). (Sofia, Bulgaria). Ed. by G. Lee. Vol. Lecture Notes in Earth Sciences. Lecture Notes in Earth Sciences. Information Engineering Research Institute. Bellflower, USA: IERI & PRESS, 2017, pp. 75–81. ISBN: 978-1-61275-526-7. DOI: 10.5729/lnes.2017.7.75 (cit. on pp. 23, 24).
- [ES09] M. Eigner, R. Stelzer. *Product lifecycle management: Ein Leitfaden für product development und life cycle management*. Springer Science & Business Media, 2009, pp. 47–62 (cit. on p. 25).
- [Fla] FlatCAM. *FlatCAM: PCB Prototyping CAD/CAM*. <http://flatcam.org/>. (Accessed on 04/13/2018) (cit. on p. 39).
- [Fre] FreeCAD. *FreeCAD: An open-source parametric 3D CAD modeler*. <https://www.freecadweb.org/>. (Accessed on 04/13/2018) (cit. on p. 38).
- [FT00] R. T. Fielding, R. N. Taylor. *Architectural styles and the design of network-based software architectures*. Vol. 7. University of California, Irvine Doctoral dissertation, 2000, pp. 76–105 (cit. on p. 28).
- [gCA] gCAD3D. *gCAD3D.org*. <http://www.gcad3d.org/>. (Accessed on 04/13/2018) (cit. on p. 38).

Bibliography

- [GITJ14] O. Givehchi, J. Imtiaz, H. Trsek, J. Jasperneite. "Control-as-a-service from the cloud: A case study for using virtualized PLCs." In: *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*. IEEE. 2014, pp. 1–4 (cit. on p. 31).
- [Hee] D. Heeks. *HeeksCAD & HeeksCNC*. <https://sites.google.com/site/heekscad/>. (Accessed on 04/13/2018) (cit. on p. 37).
- [KRS15] H. Kief, H. Roschiwal, K. Schwarz. *CNC-Handbuch 2015/2016: CNC, DNC, CAD, CAM, FFS, SPS, RPD, LAN, CNC-Maschinen, CNC-Roboter, Antriebe, Energieeffizienz, Werkzeuge, Industrie 4.0, Fertigungstechnik, Richtlinien, Normen, Simulation, Fachwortverzeichnis*. Hanser, 2015, pp. 39–56. ISBN: 9783446440906. URL: <https://books.google.de/books?id=qO7JrQEACAAJ> (cit. on pp. 25, 26).
- [MBM+12] F. Maciá Pérez, J. V. Berna-Martinez, D. Marcos-Jorquera, I. Lorenzo Fonseca, A. Ferrándiz Colmeiro, et al. "Cloud agile manufacturing." In: (2012) (cit. on p. 21).
- [MT10] N. Medvidovic, R. N. Taylor. "Software architecture: foundations, theory, and practice." In: *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*. Vol. 2. IEEE. 2010, pp. 471–472 (cit. on p. 28).
- [PyC] PyCAM. *PyCAM: a toolpath generator*. <http://pycam.sourceforge.net/>. (Accessed on 04/13/2018) (cit. on p. 38).
- [RR08] L. Richardson, S. Ruby. *RESTful web services*. "O'Reilly Media, Inc.", 2008, pp. 79–81 (cit. on p. 28).
- [RZW+17] L. Ren, L. Zhang, L. Wang, F. Tao, X. Chai. "Cloud manufacturing: key characteristics and applications." In: *International Journal of Computer Integrated Manufacturing* 30.6 (2017), pp. 501–515 (cit. on p. 30).
- [SPF+07] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, L. Peterson. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors." In: *ACM SIGOPS Operating Systems Review*. Vol. 41. 3. ACM. 2007, pp. 275–287 (cit. on p. 28).
- [SSS07] S.-J. Shin, S.-H. Suh, I. Stroud. "Reincarnation of G-code based part programs into STEP-NC for turning applications." In: *Computer-Aided Design* 39.1 (2007), pp. 1–16 (cit. on p. 26).
- [TCD+14] F. Tao, Y. Cheng, L. Da Xu, L. Zhang, B. H. Li. "CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system." In: *IEEE Transactions on Industrial Informatics* 10.2 (2014), pp. 1435–1442 (cit. on pp. 21, 30).
- [TMTR15] S. Tedeschi, J. Mehnen, N. Tapoglou, R. Rajkumar. "Security aspects in Cloud based condition monitoring of machine tools." In: *Procedia CIRP* 38 (2015), pp. 47–52 (cit. on p. 21).

- [TZV+11] F. Tao, L. Zhang, V. Venkatesh, Y. Luo, Y. Cheng. “Cloud manufacturing: a computing and service-oriented manufacturing model.” In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 225.10 (2011), pp. 1969–1976 (cit. on pp. 17, 21).
- [WGRS13] D. Wu, M. J. Greer, D. W. Rosen, D. Schaefer. “Cloud manufacturing: Strategic vision and state-of-the-art.” In: *Journal of Manufacturing Systems* 32.4 (2013), pp. 564–579 (cit. on pp. 21, 22, 30).
- [XNR+13] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, C. A. De Rose. “Performance evaluation of container-based virtualization for high performance computing environments.” In: *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. IEEE. 2013, pp. 233–240 (cit. on pp. 28, 29).
- [Xu12] X. Xu. “From cloud computing to cloud manufacturing.” In: *Robotics and computer-integrated manufacturing* 28.1 (2012), pp. 75–86 (cit. on pp. 17, 21, 22).
- [Xu17] X. Xu. “Machine Tool 4.0 for the new era of manufacturing.” In: *The International Journal of Advanced Manufacturing Technology* 92.5-8 (2017), pp. 1893–1900 (cit. on p. 30).
- [YZZB17] X. Yao, J. Zhou, J. Zhang, C. R. Boër. “From intelligent manufacturing to smart manufacturing for Industry 4.0 driven by next generation artificial intelligence and further On.” In: *Enterprise Systems (ES), 2017 5th International Conference on*. IEEE. 2017, pp. 311–318 (cit. on p. 30).

All links were last followed on May 6, 2018.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature