ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA STROJNÍ

Studijní program: N 2301 Strojní inženýrství
Studijní zaměření: 2303T004 Strojírenská technologie – technologie obrábění

DIPLOMOVÁ PRÁCE

Postprocesor pro zvolený stroj k SW FreeCAD

Autor: **Bc. Martin FRNOCH**

Vedoucí práce: **Ing. Jiří VYŠATA, Ph.D.**

Akademický rok 2017/2018

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin FRNOCH**
Osobní číslo:	**S16N0042K**
Studijní program:	**N2301 Strojní inženýrství**
Studijní obor:	**Strojírenská technologie - technologie obrábění**
Název tématu:	**Postprocesor pro zvolený stroj k SW FreeCAD.**
Zadávající katedra:	**Katedra technologie obrábění**

Zásady pro vypracování:

1. Úvod
2. Specifikace problému
3. Návrh postprocesoru
4. Ověření funkce postprocesoru
5. Závěr

Rozsah grafických prací: **dle potřeby**

Rozsah kvalifikační práce: **50 - 70 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

- **Jandečka K., Česánek J., Kožmín P.: Programování NC strojů. Plzeň: ZČU, 2000**
- **Siemens: Příručka pro osbluhu SINUMERIK 828D / 840D sl frézování. Německo: 2015.**
- **Siemens: Programovací příručka SINUMERIK 840D sl / 828D Základy. 2010**
- **Siemens: Programovací příručka SINUMERIK 840D sl / 828D Pro pokročilé. 2010**
- **KOŽMÍN,P.: Disertační práce - Metoda tvorby víceosého postprocesoru. ZČU, Plzeň: 2004.**
- **ŠTULPA, M.: CNC obráběcí stroje a jejich programování. Praha: 2006.**
- **FreeCAD wiki**
- **IT network-Python**

Vedoucí diplomové práce: **Ing. Jiří Vyšata, Ph.D.**
Katedra technologie obrábění

Konzultant diplomové práce: **Ing. Jan Hnátík, Ph.D.**
Katedra technologie obrábění

Datum zadání diplomové práce: **16. října 2017**

Termín odevzdání diplomové práce: **21. května 2018**

Doc. Ing. Milan Edl, Ph.D.
děkan

L.S.

Západočeská univerzita v Plzni * Fakulta strojní *

Doc. Ing. Jan Řehoř, Ph.D.
vedoucí katedry

V Plzni dne 18. října 2017

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.
Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů, uvedených v seznamu, který je součástí této diplomové práce.

V Plzni dne: ……. . . . . . .
podpis autora

Poděkování

Chtěl bych tímto poděkovat vedoucímu mé diplomové práce panu Ing. Jiřímu Vyšatovi, Ph. D., za velmi cenné rady při zpracovávání a celkové vedení této práce. Dále chci poděkovat panu Ing. Janu Hnátíkovi Ph. D., za rady tykající se programování a pomoc s přípravu programů. Poděkování také patří taky panu Doc. Ing. Jiřímu Česánkovi Ph. D., za poskytnutí možnosti ověření programů přímo na stroji. Na závěr bych chtěl také poděkovat své rodině a přítelkyni za podporu během celého studia.

ANOTAČNÍ LIST DIPLOMOVÉ PRÁCE

AUTOR	**Příjmení** Bc. Frnoch	**Jméno** Martin
STUDIJNÍ OBOR	2303T004 „Strojírenská technologie – technologie obrábění“	
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Ing. Vyšata Ph.D.	**Jméno** Jiří
PRACOVIŠTĚ	ZČU - FST – KTO	

DRUH PRÁCE	**DIPLOMOVÁ**	~~**BAKALÁŘSKÁ**~~	**Nehodící se škrtněte**
NÁZEV PRÁCE	**Postprocesor pro zvolený stroj k SW FreeCAD**		

FAKULTA	strojní		**KATEDRA**	KTO		**ROK ODEVZD.**	2018

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	73		**TEXTOVÁ ČÁST**	55		**GRAFICKÁ ČÁST**	18

STRUČNÝ POPIS (MAX 10 ŘÁDEK) ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY	Tato práce se zabývá tvorbou postprocesoru pro software FreeCAD. K čemuž je využit programovací jazyk Python. Zmíněný postprocesor je vytvářen pro zvolený stroj, kterým je EMCO Concept MILL 105 s řídicím systémem Sinumerik 828D. Práce si klade také jako druhotný cíl představit FreeCAD, jakožto bezplatného, open-source CAD/CAM systém a rozšířit tak povědomí o tomto software.
KLÍČOVÁ SLOVA ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE	Postprocesor, Python, CNC, FreeCAD, CAD, CAM, Open-source, ISO kód, EMCO Concept MILL 105, Sinumerik 828D

SUMMARY OF DIPLOMA SHEET

AUTHOR	**Surname** Bc. Frnoch	**Name** Martin	
FIELD OF STUDY	2303T004 „Manufacturing Processes – Technology of Metal Cutting“		
SUPERVISOR	**Surname (Inclusive of Degrees)** Ing. Vyšata Ph.D.	**Name** Jiří	
INSTITUTION	ZČU - FST - KTO		
TYPE OF WORK	**DIPLOMA**	~~**BACHELOR**~~	**Delete when not applicable**
TITLE OF THE WORK	**Postprocessor for selected machine to software FreeCAD.**		

FACULTY	Mechanical Engineering		**DEPARTMENT**	Machining Technology		**SUBMITTED IN**	2018

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	73		**TEXT PART**	55		**GRAPHICAL PART**	18

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	This thesis deals with the creation of a postprocessor for FreeCAD software. For this creation the Python programming language is used. Postprocessor is designed for the chosen machine, which is EMCO Concept MILL 105 with the Sinumerik 828D control system. Secondary goal of this thesis is also an introduction of FreeCAD as a free open-source CAD / CAM system, and thus extend awareness of this software.
KEY WORDS	Postprocesor, Python, CNC, FreeCAD, CAD, CAM, Open-source, ISO kód, EMCO Concept MILL 105, Sinumerik 828D

Obsah

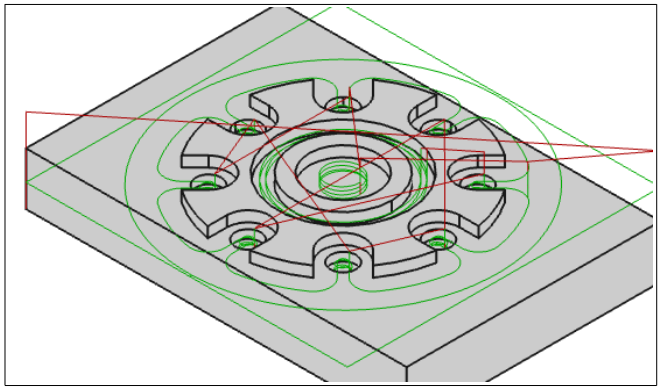
Seznam použitých zkratek a symbolů	
Zkratka	**Význam**
TPV	Technická Příprava Výroby
CAx	Computer Aided x – Počítačem podporované x (Náhrada za CAD/CAM)
CAD	Computer Aided Design – Počítačem podporovaný návrh
CAM	Computer Aided Machining – Počítačem podporovaná výroba
PLM	Product Lifecycle Management – Systém řízení životního cyklu produktu
CNC	Computer Numeric Control – Počítačové číslicové řízení
LGPL	Lesser General Public Licence – Nižší obecná veřejná licence
3D	Three Dimensional - Trojrozměrný
2D	Two Dimensional - Dvourozměrný
GPL	General Public Licence – Obecné věřejná licence
EULA	End-User-Licence-Agreement – Licenční smlouva pro koncové uživatele
CL data	Cutter Location Data – Informace o poloze řezného nástroje

Tab. 1: Seznam použitých zkratek a symbolů

1. Úvod

Ve strojním průmyslu je pro firmy jedním z důležitých faktorů, a většinou i tím nejdůležitějším, zisk. To vede majitele a jednatele těchto firem k minimalizaci nákladů. O tuto nákladovou minimalizaci se nesnaží pouze velké firmy s hromadnou či sériovou výrobou, ale také menší podniky s malosériovou nebo kusovou výrobou. Tato nákladová minimalizace může být nalezena v mnoha aspektech výroby, například změna dodavatele materiálu či pořízení výkonnějšího stroje nebo optimalizací výrobního procesu. Úsporu nákladů je možné hledat také v nehmotné stránce výroby, konkrétněji v softwarovém vybavení firmy, kam patří například informační systémy, ekonomický a komunikační software a další software používaný pro TPV[1] hlavně zaměřený na konstrukční a technologickou přípravu, tedy CAD/CAM systémy.

CAM systémy se využívají především tam, kde jsou ve výrobě nasazeny CNC stroje. Tyto systémy přinášejí řadu výhod, zejména usnadnění vytváření obráběcí technologie pro složitější obrobky. Pro ty by tvorba programu v dílenském programování u stroje byla časově náročná nebo dokonce nemožná, zejména v případě tvarových ploch. Jednou z velkých nevýhod těchto systémů je vysoká pořizovací cena a poměrně vysoké roční poplatky za technickou podporu software. Právě proto je cílem této práce vytvoření postprocesoru pro modul „Path[2]“ bezplatné aplikace FreeCAD. Tato aplikace spadá pod licenci LGPL[3] a je ve fázi vývoje, což lze poznat podle toho, že označení aktuální stabilní verze software je 0.16 a další vyvíjená verze má označení 0.17. Běžně má první publikovaná hotová verze kteréhokoliv software označení 1.0. Na obrázku (Obr. 1) je možné vidět dráhy nástroje vytvořené v Path modulu aplikace FreeCAD pro jednu ze vzorových součástí. Kde zelené křivky reprezentují pracovní posuv a červené rychloposuv. Předlouhou pro model součásti byla výkresová dokumentace ze zdroje [1].



Obr. 1 - Dráhy nástroje

Po vytvoření postprocesoru je dalším cílem ověření jeho funkčnosti, a to vygenerováním ISO kódu pro předem zvolený stroj. Tímto strojem je EMCO CONCEPT MILL 105. Jde o malou tříosou frézku, která se využívá pro účely výuky v prostorách katedry technologie obrábění fakulty strojní Západočeské univerzity v Plzni.

[1] Technická příprava výroby – Konstrukční příprava, Technologická příprava, Technicko-organizační příprava

[2] Označení modulu, které má napovědět, že jde o tvorbu drah nástroje při obrábění, anglické slovo Path znamená cesta nebo dráha. Další informace v kapitole 3.4.3

[3] Lesser General Public Licence – Nižší Obecná Veřejná Licence (Vysvětleno dále v 2.3)

2. Specifikace problému

Na začátku je vhodné si ujasnit všechny vstupní informace a okolnosti související s řešením problematiky tvorby postprocesoru. Nejprve se musí vyjasnit cíl práce a poté následuje specifikace použitých činitelů pro samotnou realizaci. Zmíněnými činiteli jsou konkrétně FreeCAD, Python a školní frézka EMCO CONCEPT MILL 105.

2.1. Vysvětlení podstaty a cíle

Pro jasné pochopení podstaty cíle práce je nutné si nejprve zodpovědět několik otázek. První z nich je, proč se zabývat právě tvorbou postprocesoru. Na tuto otázku je hned několik odpovědí. První je, že tato oblast si zaslouží podrobnější prozkoumání neboť postprocesing, tedy překlad výstupu CAM systému na funkční vstup pro obráběcí stroj, paří mezi klíčové kroky při realizaci obrábění z hlediska automatizace. Druhou odpovědí, která zároveň odpovídá i na další otázku, proč byl zvolen právě již zmíněný software FreeCAD je, že povědomí o existenci tohoto software není příliš vysoké. Většina uživatelů a to jak pro profesní, tak i pro soukromé účely zvolí raději komerční varianty. Vzniká zde určitá averze proti těmto bezplatným software. Například na katedře se používají z CAx systémů hlavně CATIA a SolidWorks. Práce si tak také klade nepřímo za cíl, kromě příspěvku k tématice tvorby postprocesoru, i představení této bezplatné aplikace. Je jednoznačné, že FreeCAD se v současné době s těmito komerčními velikány v oboru nemůže v žádném případě měřit. Na druhou stranu jde ale o bezplatný software, který je stále ve vývoji a pomalými krůčky se stává stále více sofistikovanějším. Zpravidla open-source[4] software začíná jako podřadná napodobenina té nejpoužívanější a nejznámější komerční varianty. Příkladem pro srovnání zde může být kancelářský komerční software Office Word od společnosti Microsoft a bezplatný Open Office (Libre Office) Writer od The Document Foundation. V dnešní době je Writer minimálně na stejné úrovni jako komerční Word z hlediska funkčnosti a v některých případech ho dokonce překonává svými dostupnými funkcemi. Mezi které patří například typografické úpravy.

Pro to aby bylo možné FreeCAD využívat nejen pro modelování a vytváření technologie obrábění, ale i pro samotnou výrobu je nezbytný právě postprocesor, který převede výstup FreeCADu na použitelná data pro zvolený CNC stroj. Tento proces je vysvětlen podrobněji v následující kapitole 3.

V pořadí třetí otázkou, kterou je nutno zodpovědět je, proč se používá právě programovací jazyk Python. Jednoznačnou odpovědí je, že FreeCAD disponuje python konzolí a je možné ho díky ní přímo ovládat a také dále rozšiřovat. Většina rozšíření FreeCADu je vytvořena právě v Pythonu. Stejně tak všechny postprocesory překládající výstupní data FreeCADu jsou také vytvořeny v Pythonu.

Poslední čtvrtou otázkou je, proč právě stroj EMCO CONCEPT MILL 105. Hlavním důvodem je, že na fakultě je k dispozici a to i pro studenty ve výuce. Pokud FreeCAD nezanikne jako mrtvý projekt, což je bohužel někdy smutnou realitou u těchto open-source projektů, naskýtá se zde do budoucna možnost používání FreeCADu na katedře spolu s komerčními CAx systémy a rozšířit tak díky tomu, povědomí o této aplikaci.

[4] Software s otevřeným zdrojovým kódem z anglického open-source, vysvětleno v kapitole 2.3

V případě, že by došlo v průběhu práce k nalezení postprocesoru FreeCADu pro stroj EMCO CONCEPT MILL 105 ať už při rešerši či při samotné tvorbě postprocesoru nebo dokonce finalizaci práce, je stále nutné proces tvorby postprocesoru zmapovat a vytvořit tak základ pro další možné zkoumání této problematiky. Proto je práce koncipována tak, aby pro případné budoucí tvorby postprocesoru, právě pro FreeCAD mohla být užitečnou pomůckou. Dalším nepřímým cílem práce je také propagace FreeCADu.

2.2. Popis činitelů s prací souvisejících

Na tuto práci mají vliv celkem tři okolnosti, kterým je dobré se věnovat a zorientovat se tak alespoň rámcově jakým způsobem v práci figurují. S tím, že některé z nich pravděpodobně bude potřeba dále zevrubněji prostudovat v rámci příslušných analýz.

Obr. 2: - FreeCAD logo [2]

Prvním z představovaných činitelů a patrně základním, jak vyplývá již ze zaměření práce, je software FreeCAD. Jde o bezplatný open-source CAx systém, ostatně jak již bylo zmíněno výše. Ten byl zpočátku používaný především pro parametrické modelování. Software je šířen pod licencí LGPL, která je vysvětlena v následující podkapitole 2.3. Vzhledem k tomu, že cílem práce je vytvoření postprocesoru pro zvolený stroj, tak bude nutné se nejprve seznámit se způsobem práce s tímto systémem. Tedy s tvorbou modelů a s vytvářením drah nástroje, což může být pravděpodobně podobné s komerčními CAx systémy typu CATIA, SolidWorks a dalšími méně známými variantami. Dále je nezbytné se seznámit se způsobem fungování postprocesorů, a především se způsobem tvorby postprocesorů pro tento systém.

Obr. 3: Python logo [3]

Právě pro tvorbu (programování) postprocesorů slouží programovací jazyk Python, jenž je dalším činitelem práce. Figuruje v práci především z toho důvodu, že postprocesory FreeCADu jsou v tomto programovacím jazyce vytvářeny. Stejně jako další rozšíření FreeCADu. Zatím není jiný způsob jak postprocesor získat, neboť FreeCAD nedisponuje v současné době generátorem postprocesorů. Bude tak nutné se seznámit se způsobem práce a vytvářením skriptů v tomto programovacím jazyce, aby bylo možné vytvořit postprocesor pro zvolený stroj pomocí ručního vytváření kódu a nikoliv vytvoření postprocesoru pomocí generátoru. Úkolem tohoto kódu je převedení výstupu z FreeCADu, tedy interního kódu drah nástroje na ISO kód, tedy kód kompatibilní s řídicím systémem zvoleného cílového stroje.

Tímto cílovým strojem je EMCO CONCEPT MILL 105, což je i třetí činitel práce. Jde o školní tříosou frézku používanou pro účely výuky v učebně UL 211. Jedná se o stroj s poměrně malým pracovním prostorem a stejně tak i jeho výkonem. Podrobnější popis stroje bude následovat v příslušné analytické části. V průběhu tvorby práce se očekává změna řídícího systému a to z emulátoru řídicího systému Sinumerik 840D na plnohodnotný řídicí systém Sinumerik 828D. Předpokládaným systémem, pro který má být postprocesor vytvořen, je tedy Sinumerik 828D. Od ručně psaného postprocesoru se

očekává, že bude využívat maximální potenciál stroje a řídicího systému. V tomto případě zde ještě může nastat omezení ze strany samotného CAM systému, který nemusí zatím podporovat všechny funkce, jimiž řídící systém na stroji disponuje, neboť jde stále zatímo software ve fázi vývoje. V článku Miroslava Sadílka totiž odpovídá Ing. František Srovnal na otázku, jak hodnotí roli postprocesoru v přípravě technologie: „*Role dobře naprogramovaného postprocesoru je naprosto klíčová, postprocesor musí mít v sobě zapracovány veškeré funkce stroje, aby bylo možné stoprocentně využít jeho potenciál"*.[4]

2.3. Legislativní stránka problematiky

Je vhodné si položit otázku proč vůbec má smysl zabývat se otázkou legislativní problematiky. Očekává se, že každá diplomová práce bude obsahovat technicko-ekonomické zhodnocení svých výstupů. Do ekonomického zhodnocení v tomto případě vstupuje jedna zvláštnost, která souvisí s FreeCADem. Jde o to, že tento software je vyvíjen v licenci LGPL a ačkoliv je zatím ve vývojové fázi, očekává se, že tato fáze jednou překlene do ostré a teprve pak, se z tohoto software může stát zajímavá alternativa komerčních software, kterou je možné porovnávat s ostatními. V současném stavu je vhodné namísto porovnání prozkoumat spíše dosavadní možnosti tohoto software. Dále je vhodné upozornit na skutečnost, že je volně přístupný, což se může stát zajímavým vedlejším efektem práce. S tím souvisí skutečnost, upřesnit co vlastně legislativní stránka problematiky znamená. Technické hodnocení, které porovnává výchozí a konečný stav, může být součástí kapitoly shrnující výsledky práce.

Jak již bylo několikrát zmíněno, FreeCAD je open-source software vyvíjen v licenci LGPL. To má své významné důsledky. Open-source software znamená, že jeho zdrojový kód je volně přístupný a uživatel ho může podle svých potřeb měnit a volně šířit, pokud to umožňuje licence, pod kterou je publikován. V případě licence LGPL, jejíž podstata bude vysvětlena následovně, to pro běžného uživatele znamená, že software muže získat zdarma a také ho zdarma či za poplatek beztrestně šířit. Poplatek je v tomto případě za cenu šíření, například za datové medium, nikoliv za software, který je zdarma. Kromě toho to znamená, že je vyvíjen dál průběžně komunitou nadšenců a běžný uživatel může počítat s tím, že FreeCAD bude vyvíjen dále.

Licence LGPL a GPL mají svá zvláštní specifika, která software ve společnosti uživatelů poskytují určité postavení a určitou životnost. To souvisí také mimo jiné s jedním z cílů této práce, který byl již zmíněn výše, tedy propagovat tento software. Proto je zde užitečné tuto zajímavou licenci představit, neboť z toho jak funguje mohou plynout důsledky pro to, jaká je pravděpodobnost dalšího vývoje software.

Materiály k této tématice jsou dostupné na webových stránkách gnu.org[5]. Zde jsou materiály zpravidla dostupné pod licencí creative commons, kde velmi často u článků není uveden jeho konkrétní autor. Nicméně vždy když se v textu použije nějaká citace ze zdroje s licencí creative commons, bude vždy uveden zdroj.

Je vhodné také vysvětlit co je to free software, kde je v tomto případě myšleno jako spíše svobodný software než zadarmo. Na stránkách gnu.org je vysvětleno, jaká je filozofie svobodného software, aby byl program svobodným software musí mít čtyři základní svobody.[5]

Svoboda 0 – Spustit program za jakýmkoliv účelem.

Svoboda 1 – Studovat jak program funguje a měnit ho podle svých potřeb. K tomuto je podmínkou mít přístup k zdrojovému kódu programu.

Svoboda 2 – Šířit kopie programu.

Svoboda 3 – Šířit své upravené verze programu aby mohla celá komunita využít výhod vašich změn. Zde je opět podmínka přístupu ke zdrojovému kódu.

Opakem svobodného software je proprietární (účelový), který autor poskytuje za poplatek, a jeho šíření i upravování je zakázáno. Uživatel má tak povinnost při instalaci tohoto software souhlasit s podmínkami smlouvy EULA, tedy „End-User-Licence-Agreement“, což je smlouva pro koncové uživatele software, kde se uživatel zavazuje k dodržení podmínek stanovených poskytovatelem. Většinou jde mimo jiné o zákaz šíření a editaci instalovaného software. Uživatel kupuje vlastně právo na používání software podle daných pravidel, nikoli software samotný.

Na stránkách továrna.cz je licence GPL, jejíž akronym „General Public License“ znamená „Obecná Veřejná Licence“, specifikována jako licence, která vyžaduje, aby všechna díla používající dílo pod licencí GPL spadala rovněž pod licenci GPL. Naproti tomu zmíněná licence LGPL je méně přísnou formou licence GPL. Akronym LGPL znamená „Lesser General Public License“, což lze volně přeložit z anglického jazyka jako „Nižší Obecná Veřejná Licence“. Touto zkratkou se označuje licence svobodného (volného) software, díky které je možné použít již vytvořený software (kód) jako součást jiného díla, jež nemusí být pod licencí LGPL. Tím je myšleno propojení s dalším software, které nemodifikuje samotné původní dílo pod licencí LGPL. Typicky se jedná o softwarové knihovny, které je povoleno používat i v komerčních projektech. To tedy znamená, že díky této licenci je možné na vytvořených projektech pro FreeCAD i profitovat. Ovšem komunita uživatelů je velmi solidární a své výtvory poskytuje ve většině případů zdarma. Velmi často jde také o vzájemnou spolupráci více uživatelů. Zjednodušeně lze tedy říci, že GPL zajišťuje rozvoj svobodného software, zatímco LGPL umožňuje komerční využití výsledků svobodného software. [6]

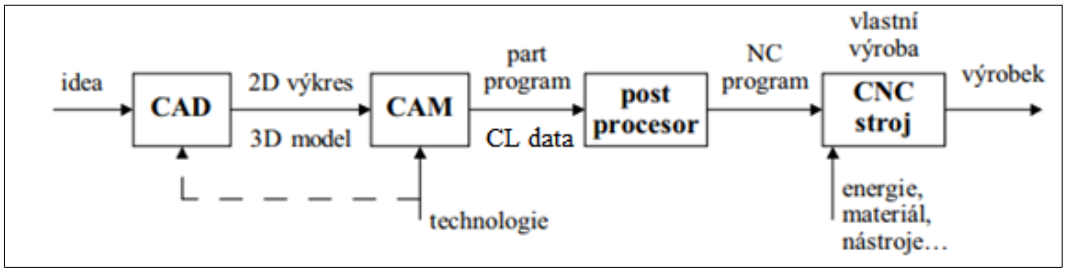
3. Analýza stávající situace

Po specifikaci problematiky práce a upřesnění jejího cíle je ještě nutné, před samotným vytvářením postprocesoru a detailním popisem jeho tvorby, vyjasnit fakta a poznatky z této problematiky. Tedy podrobně analyzovat veškeré aspekty vstupující do tohoto tématu. Sem patří oblasti rešeršního charakteru, a to sice problematika postprocesorů jako taková a dále průzkum dosavadních výsledků. Poté následuje detailní popis FreeCADu, a to zejména jeho obráběcího Path modulu. Dalším bodem je seznámení s používáním Pythonu, tedy především způsobu programování v tomto programovacím jazyce. Na závěr je popsán stroj včetně ISO kódu.

3.1. Analýza problematiky postprocesorů

Vzhledem k tomu, že hlavním cílem práce je vytvoření postprocesoru pro frézku EMCO CONCEPT MILL 105 a to pomocí programovacího jazyka Python ve spojení se software FreeCAD, ostatně jak již bylo zmíněno, je nutné nejprve si ujasnit, co to vlastně postprocesor je, a jaká je jeho funkce. Postprocesor je podle Miroslava Sadílka: „*Softwarový převodník dat z CAD/CAM systému do datového jazyka konkrétního obráběcího stroje. Kvalitní postprocesor v sobě obsahuje veškeré informace o vlastnostech daného stroje, tak aby bylo optimálně a efektivně využito všech jeho funkcí v souladu s CAD/CAM systémem. Univerzální postprocesor neexistuje, bohužel je potřeba jej naprogramovat pro každý stroj zvlášť. Cesta k dokonalému postprocesoru se stává obtížnější i kvůli faktu, že pravděpodobně neexistují dvě identické konfigurace obráběcího CNC stroje. Počet možných kombinací je velmi vysoký. Když toto uvážíme, není překvapující vysoká možnost nezdaru při neodborném programování*.“[4]

Pro funkční výstup z CAM systému, je tedy nutné mít správný postprocesor pro daný řídicí systém cílového stroje. Na diagramu (Obr. 4) je možné vidět proces výroby pomocí NC programu s použitím CAD/CAM systému, ve kterém hraje postprocesor důležitou roli. Popišme si nyní diagram z levé strany. Nejprve je na základě nápadu vytvořen 3D model v CAD systému, a to i včetně výrobní dokumentace. Model je následně přenesen do CAM systému. Zde se vytvoří technologie obrobení pro součást, tedy part program a CAM systém vygeneruje CL data[5]. Tato data jako taková nejsou použitelná pro CNC stroj, proto je postprocesor převádí do čitelné podoby pro daný stroj. CAM systém a postprocesor jsou jeden bez druhého nepoužitelné a vzájemně se doplňují. Výstupem převodu je žádaný NC program, který CNC stroj dokáže přečíst. Díky tomu může, po dodání materiálu, energie a nástrojů proběhnout finální proces, kterým je vlastní výroba součásti.



Obr. 4 Výrobní diagram s použitím CAM systému [7]

[5] CL data (Cutter location data) – dále viz. 3.4.3.1

Již zmíněná CL data tedy Cutter Location Data, neboli informace o poloze řezného nástroje, jsou výstupem z CAM systému. Tento výstup obsahuje informace nezbytné pro pohyb nástroje, ovšem pro cílový stroj je nepoužitelný, neboť má úplně odlišnou syntaxi. Pro přeložení této syntaxe na čitelnou podobu použitelnou pro stroj, slouží právě postprocesor.

Postprocesory je možné rozdělit na interní a externí. Interní postprocesory jsou součástí CAM systému a zpravidla jsou dodávány společně s ním. Jejich externí protějšky jsou samostatné aplikace pro přeložení jakéhokoliv druhu výstupu (CL dat) na specifický kód stroje. Postprocesory je také možné rozdělit podle způsobu jejich vzniku. Tedy buď je postprocesor vytvořen ručně v různých programovacích jazycích (Java, C++, Python) a nebo pomocí generátoru postprocesorů, který bývá součástí CAx systémů.

Ručně psané (programované) postprocesory jsou vytvářené pro specifický stroj a je brán zřetel na jeho funkce, zde je ovšem nutná znalost programovacího jazyka a funkce NC strojů. Na vytváření ručně programovaných postprocesorů existují specializované firmy, které tyto postprocesory dodávají. Zde je možné se setkat s bohužel obvyklým problémem z praxe, kdy dodávka požadovaného postprocesoru není dle prvotních přání zákazníka. Ať už jde o dobu dodání, která se může prodloužit i na několik let, či funkčnost dodaného postprocesoru, která neodpovídá smluveným požadavkům. To může vést v extrémních případech až k soudnímu jednání. Druhou možností je, že si postprocesor naprogramuje sám uživatel CAM systému, zde ale mohou vznikat problémy z hlediska samotného fungování postprocesoru. To závisí na programovacích schopnostech uživatele. U postprocesoru vygenerovaného pomocí generátoru je vytvoření otázkou vyplnění dotazových tabulek generátoru. Celý proces je tak velmi zjednodušený, ovšem nelze počítat s postprocesorem na míru pro každý stroj.

Vhodnost volby typu postprocesoru je okolností, která komplikuje situaci, tedy buďto je nutné se spokojit s univerzálním postprocesorem, kde je možné a vcelku pravděpodobné očekávat chyby či nepřesnosti ve vygenerovaném výstupu pro cílový stroj, a tento výstup je pak následně nutné ručně upravit do konečné fáze. Další možnost je zainvestovat do specifického postprocesoru či se ho pokusit vytvořit. Ovšem ten je vytvářený se zaměřením pouze pro daný stroj a při použití na jiném stroji můžou vznikat problémy. Postprocesor lze tak považovat za problémové místo při získaní kvalitního výstupu z CAM systému. Zejména v případě jde-li o bezplatné CAM systémy, které jsou na tom o poznání hůře, než komerčně používaný software. Zde není možné počítat s dodaním postprocesoru na míru od specializované firmy a zbývá pak jen buďto se spokojit s postprocesory dostupnými ve veřejných knihovnách, a nebo se pokusit o vlastní tvorbu.

Z odborných zdrojů, které se obecně zabývaly problematikou postprocesorů byly nalezeny následující publikace. První z nich je bakalářská práce J. Lercha s názvem „Tvorba postprocesoru pro sinumerik 840 ve vybraném CAD/CAM systému“. Cílem této práce byla tvorba postprocesoru dvěma způsoby také právě pro stroj EMCO Concept MILL 105. Jedním bylo použití generátoru postprocesorů systému EdgeCAM a druhým volné programování v systému Cimatron^{it} v jazyce podobném systému Fortran. Z této práce lze využít obecné informace tykajících se postprocesorů ale především zdroje týkající se informacích o stroji, ale bohužel již nikoliv informace o řídicím systému, který bude nahrazen Sinumerikem 828D.[8]

Další publikací je disertační práce Dr. Kožmína s názvem „Metoda tvorby víceosého postprocesoru“. Ta se zabývala vytvořením víceosého postprocesoru v programovacím jazyce C pro CAx systém CATIA. Tato práce je využita zejména jako inspirace pro popis výstupních dat ze systému CATIA a jejich porovnání s výstupem FreeCADu. [9]

3.2. Studie současné situace v dané oblasti

Jedním z důležitých kroků před samotným vytvářením postprocesoru je, získat konkrétní poznatky a informace z této problematiky a případně zjistit, zda již požadovaný postprocesor nebo jemu podobný, nebyl vytvořen. Zde je jistou nevýhodou fakt, že čerpané informace jsou velmi často získávány z komunitního fóra aplikace FreeCAD, a dostupné rozpracované dokumentace této komunity, a zpravidla nejsou dostupné žádné jiné informace týkající se problematiky postprocesorů pro FreeCAD.

Komunitní fórum na adrese https://forum.freecadweb.org/ je místem, kde uživatelé přispívají novými poznatky, problémy na které narazili při užívání, a také návrhy na zlepšení. Mezi těmito příspěvky lze nalézt i mnohá rozšíření a vylepšení od těchto samotných uživatelů, která lze ihned zakomponovat do FreeCADu. Vývojáři FreeCADu zde uvádějí změny nových verzí, a schválené rozšíření od dalších uživatelů, které jsou součástí nových verzí.

Ucelené informace technického charakteru jsou tedy k dispozici pouze v dokumentaci FreeCADu a na komunitním fóru uživatelů. Dokumentace typu technické zprávy, která by mohla být použita jako odborná publikace, například kvalifikační práce či článek v odborném časopise, bohužel nebyla nalezena. Důvodem jsou nedostatečné zdroje, které by se touto problematikou zabývaly. To může být také zapříčiněno nízkým povědomím o tomto software. Vedlejší úlohou této práce je také tuto skutečnost změnit.

Vývojová verze FreeCADu 0.17 s číslem revize 12570 obsahuje k datu 26.11.2017 celkem 11 postprocesorů. Všechny tyto postprocesory lze po instalaci FreeCADu nalézt na pevném disku na následující cestě:

[Instalační složka FreeCAD/Mod/Path/PathScripts/Post]

Sem patří i nově vytvořené postprocesory. Všechny postprocesory je možné poznat podle toho, že se název postprocesoru skládá z vlastního specifického označení následovaného podtržítkem a řetězcovou konstantou post zakončeným typem souboru.py, který naznačuje že jde o kód vytvořený v pythonu. Například „linuxcnc_post.py“ je postprocesor pro řídící systém LinuxCNC. V následující tabulce (Tab. 2) je seznam a krátký popis momentálně dostupných postprocesorů, které jsou dodávány spolu s instalačními daty FreeCADu verze 0.17 s číslem revize 12570.

Postprocesor (_post)	Využití
Centroid	centroid (3 osá férzka)
Dumper	„Surový“ výstup z Path modulu – CL data
Dynapath	Stroj - Tree Journyman 325 (tříosá frézka, ŘS - Dynapath 20)
Example	Vzorový postprocesor FreeCADu
GRBL	GRBL – ovladač krokových motorů pro CNC stroje určený pro platformu Arduino [10]
Linuxcnc	LinuxCNC (Open-source řídící systém)
Opensbp	OpenSBP (řidící systém - Shopbot)
Phillips	Stroj - Maho M 600E mill
Rml	Stroje - Roland Modela MDX
Smoothie	Smoothieboard

Tab. 2: Dostupné postprocesory FreeCADu

Po odzkoušení dostupných postprocesorů byl vybrán jako nevhodnější pro předlohu při vytváření postprocesor linuxCNC. Je to z toho důvodu, že z tohoto postprocesoru vychází i většina ostatních dostupných postprocesorů a uživatelé si ho dle svých potřeb a možností dále upravují. Mezi upravené postprocesory vycházející z postprocesoru linuxCNC patří Dynapath, GRBL a Smoothie. Postprocesory které produkují nevhodný výstup zcela odlišný od ISO kódu jsou Rml a Opensbp. Bylo neblahým zjištěním, že ostatní postprocesory produkují buďto nevhodný výstup nebo nejsou plně funkční. To je zapříčiněno změnou koncepce Path modulu, při přechodu na verzi 0.17. Kromě linuxCNC je druhým významným postprocesorem „dumper[6]_post“, který generuje z Path modulu nedotčený kód systému, tedy CL data. Ta jsou klíčovou informací pro vytvoření postprocesoru.

Při hledání informací k problematice postprocesorů vytvořených pro FreeCAD bylo neblahým zjištěním, že uživatelé se o tvorbu postprocesoru pokouší, ovšem z žádného neexistuje technická zpráva, kterou by bylo možné použít jako odrazový můstek nebo publikaci. Jediným zdrojem k problematice FreeCADu a jeho postprocesorů, se ukázala být nekompletní dokumentace FreeCADu[11], samotné skripty postprocesorů a komunitní fórum.

3.3. Použitá výkresová dokumentace

Kromě publikací zabývajících se problematikou postprocesorů je v práci použita ještě bakalářská práce Lukáše Vintra s názvem „Tvorba vzorových příkladů pro výuku NC programování pro nerotační součásti“. V této práci vytvořil student celkem dvacet součástí na výuku NC programování pro studenty. Z nichž byly vybrány čtyři součásti a jejich výkresy byly převzaty jako podklad pro vytvoření modelů vzorových součástí. Důvodem volby těchto výkresů byl především fakt že součásti byly vytvářeny právě pro stroj EMCO CONCEPT MILL[1].

[6] Z anglického slova *dump* - lze přeložit jako vypsat

3.4. FreeCAD

Jak již bylo zmíněno, FreeCAD je jedním ze stěžejních činitelů souvisejících s cílem této práce, tedy s tvorbou postprocesoru. Je proto naprosto nezbytné tento software blíže představit. Zejména přiblížit jak funguje jeho uživatelské rozhraní a základní ovládání. Dále je třeba alespoň okrajově popsat modelovací část, tedy Part a PartDesign moduly. Především je však nezbytné, se zaměřit na obráběcí modul nazývaný Path. Ten generuje dráhy nástroje a CL data. Všechny tyto informace jsou základním vstupem pro to, aby vůbec bylo možno zahájit tvorbu postprocesoru. Ještě je nutné upřesnit, že zmíněné moduly jsou ve skutečnosti workbenche[7], nebo-li pracovní prostředí. Název modul je tak v tomto případě nepřesný. Jde totiž o pracovní prostředí, tedy workbench, který je v podstatě uspořádáním grafického rozhraní a uživateli nabízí funkce pro požadovanou práci. Příkladem může být, že konstrukční workbench neobsahuje funkce pro tvorbu drah nástroje. To tedy znamená všechny funkce jsou v programu již k dispozici, ale z hlediska přehlednosti jsou dostupné jen při zvolení správného workbenche. Modulem se však považuje jakékoliv rozšíření softwarové části, které obohacuje svými funkcemi původní software. Modul je třeba samostatně doinstalovat a nemusí být součástí základního software. Dalo by se říci že každý přidaný modul obsahuje svůj vlastní workbench. Bylo by vhodné používat pojem pracovní prostředí, ale vzhledem k tomu, že docházelo ke vzniku pojmů jako „Pracovní prostření obrábění“ je v této práci workbench nebo-li pracovní prostředí, nahrazováno za modul z důvodů kompaktnosti a lepší přehlednosti textu.

Je pozoruhodné, že FreeCAD byl vytvořen již v roce 2001 Jürgenem Riegelem, jakožto prostorový parametrický modelář pro vytváření technických objektů. Protože to je open-source software, je velmi přizpůsobitelný dle požadavků uživatele, neboť jeho komunita stále přichází s dalšími vylepšeními. Mimo to si jej každý uživatel může dále rozšiřovat sám pomocí programovacího jazyka Python, pokud tento jazyk ovládá. To může spočívat v doprogramování různých funkcionalit a nebo také například tvorbou postprocesoru, jako je tomu v tomto případě. Dále je vhodné zmínit, že jde o multi-platformní software, což znamená že je možné jej používat na stejné úrovni ve všech operačních systémech tedy nejen ve Windows ale i na Linux a MacOS.

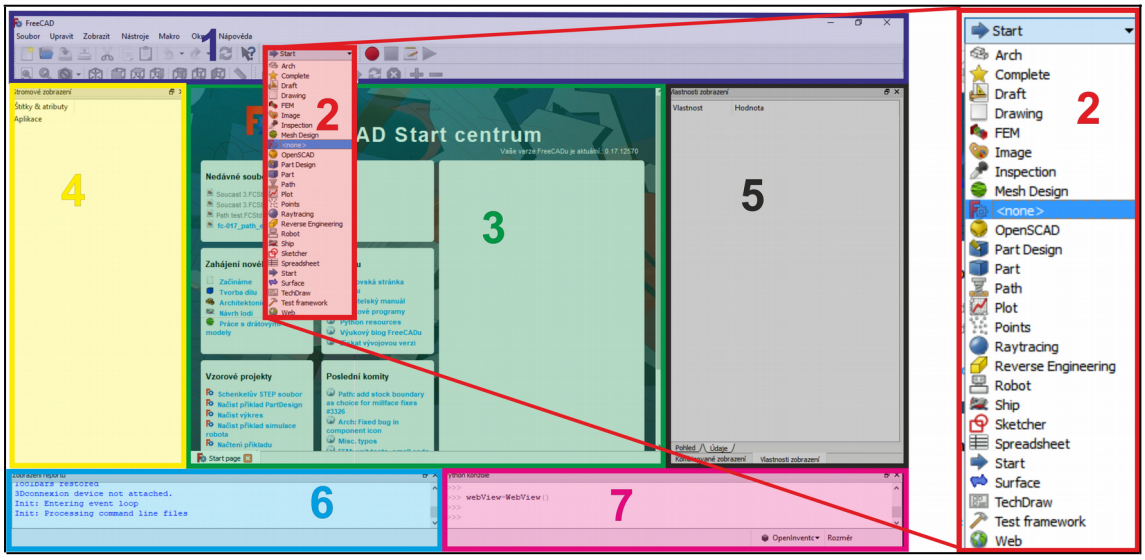
Pro tuto práci byla použita vývojová verze FreeCADu 0.17 s číslem revize 12570, která bude pravděpodobně v roce 2018 oficiálně vydána. S největší pravděpodobností dojde v budoucnu k vydání další verze, kde může dojít k markantním změnám v právě použitých modulech či dokonce samotnému ovládání. To může značně ovlivnit validitu následujícího textu. Příkladem může být předešlá verze 0.16 s revizí 6706, kde byl modul Path představen a obsahoval pouze základní funkce a přechodem na novější verzi 0.17 došlo k razantním změnám v samotné koncepci používání Path modulu, a také přidání některých experimentálních funkcí. Tyto experimentální funkce zatím bohužel nemají vytvořenou dokumentaci a uživatel se tak musí sám dovtípit jak je správně použít. Navíc u této novější zatím oficiálně nevydané verze ve fázi vývoje, není zajištěna stabilita a funkčnost těchto i stávajících funkcí. Nicméně způsob práce s různými CAD/CAM systémy má v zásadě obdobný princip. Ten, kdo se seznámí s ovládáním a koncepcí zde popisované verze na základě zde přítomného textu a bude chtít získané informace aplikovat na následující

[7] Workbench – z anglického jazyka - Pracovní prostřední (Pracovní stůl)

verze, kde může dojít ke změně způsobu postupu práce, tak to nebude s největší pravděpodobností činit výraznější potíže.

3.4.1. Základní ovládání a uživatelské rozhraní

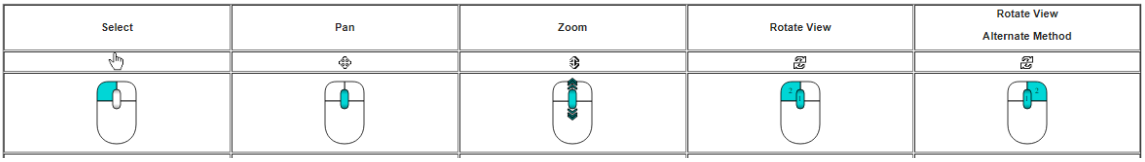
Při spuštění FreeCADu je uživateli představena úvodní obrazovka systému, kterou je možné vidět na obrázku (Obr. 5). Vyobrazené rozmístění panelů nemusí při prvotním spuštění nezbytně vypadat stejně jako na obrázku, kde již proběhla určitá úprava podle preferencí uživatele. Zmíněné rozložení lze rozdělit celkem na sedm oblastí. Ve fialovém rámečku označeném číslem 1, lze vidět lištu s příkazy, která se mění pro právě zvolený modul. Součástí tohoto rámečku je i volba modulu, která je označená červeným rámečkem a číslem 2 a její přiblížený detail je v pravé části obrázku. Zelený rámeček s číslem 3 obsahuje Start centrum. Jde o úvodní obrazovku, kde jsou odkazy na poslední používané soubory, návody a poslední úpravy aplikované od vývojářů. Při vytvoření nového souboru se tato oblast změní na trojrozměrný prostor ve kterém je pak zobrazen například vytvářený model nebo dráhy nástroje. Rámečky 4 a 5 jsou stromové a kombinované zobrazení. Zde je možné nastavit další parametry vybraných příkazů a také vidět seznam použitých funkcí na zvoleném objektu ve stromovém stylu. Modrý rámeček číslo 6 je zobrazení reportu, kde se zobrazují hlášení programu například změna modulu, výpis z python konzole nebo chybové hlášení. Poslední částí je rámeček růžové barvy s pořadovým číslem 7, kde se nachází Python konzole. Zde je možné vidět všechny právě provedené úkony v programovacím jazyce Python a také FreeCAD přímo ovládat pomocí zadávání příkazů v tomto jazyce.



Obr. 5: Úvodní obrazovka FreeCADu

Pro ovládání FreeCADu potřebuje uživatel myš a klávesnici, případně dotykovou obrazovku. Myší vybírá jednotlivé funkce a klávesnicí zadává parametry potřebné pro vybrané funkce, což je standardní způsob práce v těchto systémech. Pro práci s 3D prostorem je zde k dispozici několik stylů ovládání, respektive režimu myši *(Mouse mode)*. Na obrázku (Obr. 6) je možné vidět systém základních povelů pro navigaci v prostoru s výchozím režimem nastavení myši, který se nazývá CAD a je obdobný jako systém ovládání v CAx systému CATIA. Levým tlačítkem myši je možné vybírat například plochy nebo křivky. Přidržením kolečka myši je možné provádět posuv v rovině.

Pootočením kolečka myši je možné přibližovat a oddalovat obraz. Rotace obrazu je možná pomocí dvojhmatu, který je typický pro systém CATIA a to stisknutím a přidržením tlačítka myši v kombinaci s buďto levým nebo alternativně s pravým tlačítkem myši a následně puštěním kolečka. Uživatel si muže také zvolit jeden z dalších ovládacích stylů pokud mu tento přednastavený CAD styl nevyhovuje. Mezi tyto další styly patří OpenInventor, Bledner, Touchpad, MayaGesture a OpenCasade. FreeCAD dokonce nabízí i podporu dotykových obrazovek s režimem navigace pomocí gest *(Gesture navigation)*. Veškeré informace o povelech a dalších stylech ovládání myši jsou k dispozici v části dokumentace tykající se této oblasti viz zdroj [12].



Obr. 6: Povely myši pro navigaci v 3D prostoru [12]

FreeCAD obsahuje celou řadu zajímavých modulů. Jejich výčet je vidět v pravé části obrázku (Obr. 5) jako detail červeného rámečku s číslem 2. Zde jsou jednotlivé moduly seřazeny abecedně. Ovšem následující výčet pouze některých vybraných modulů je podle důležitosti z pohledu strojírenství a zaměření této práce. Pro tvorbu modelů ze skic souží PartDesign. Další modul pro práci s vytvořenými modely je Part, který vytváří primitivy, což jsou jednoduché geometrické tvary, a dále tyto tvary modifikuje pomocí logických vazeb mezi nimi. K obrábění je možné využít modul Path, jenž generuje dráhy nástroje. Pro výpočtové metody lze využít modul FEM[8]. Ve strojírenské praxi se stále více uplatňuje metoda reverzního inženýrství a právě k tomu má sloužit v budoucnu modul Reverse engineering, který je zatím nefunkční a slouží pouze pro testovací účely. Posledním zmíněným modulem je Robot. Ten je určen k simulaci průmyslových šestiosých robotů.

Práce se ovšem zaměří pouze na dvě oblasti. Jednou z nich je konstrukční oblast, která bude prozkoumána pouze do takové míry aby bylo možné vytvořit vzorové součásti. Do této oblasti patří moduly Part a PartDesign. Druhou oblastí je tvorba drah nástroje, k čemuž slouží modul Path. Ten kromě tvorby drah nástroje generuje i CL data, která jsou nezbytné pro tvorbu postprocesoru. Proto je prozkoumán podstatně detailněji než konstrukční moduly.

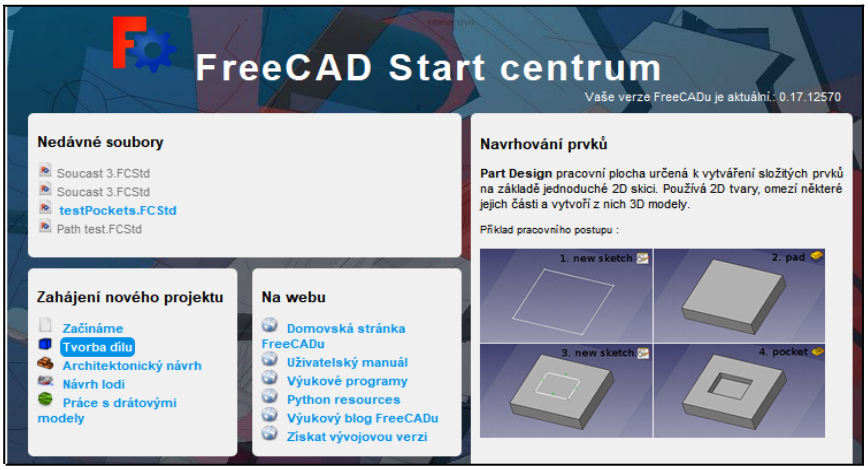
3.4.2. Moduly pro tvorbu součásti

Jedním z modulů pro tvorbu součásti je PartDesign. Ten se používá k vytváření modelů ze skic pomocí funkcí vysunutí, rotace a dalších funkcí pracujících s dvourozměrným náčrtem. Druhým je Part modul který se zaměřuje na práci s tzv. primitivními objekty.

Standardně se ve většině CAD systémů začíná s tvorbou modelu tak, že se pomocí prostorových funkcí promítne dvourozměrný náčrt, kterému se také často říká skica. Ta je promítnuta na zvolenou rovinu, která muže být buďto počátek souřadného systému anebo plocha již vytvořené části modelu. Tento postup je obdobný i v PartDesign modulu FreeCADu.

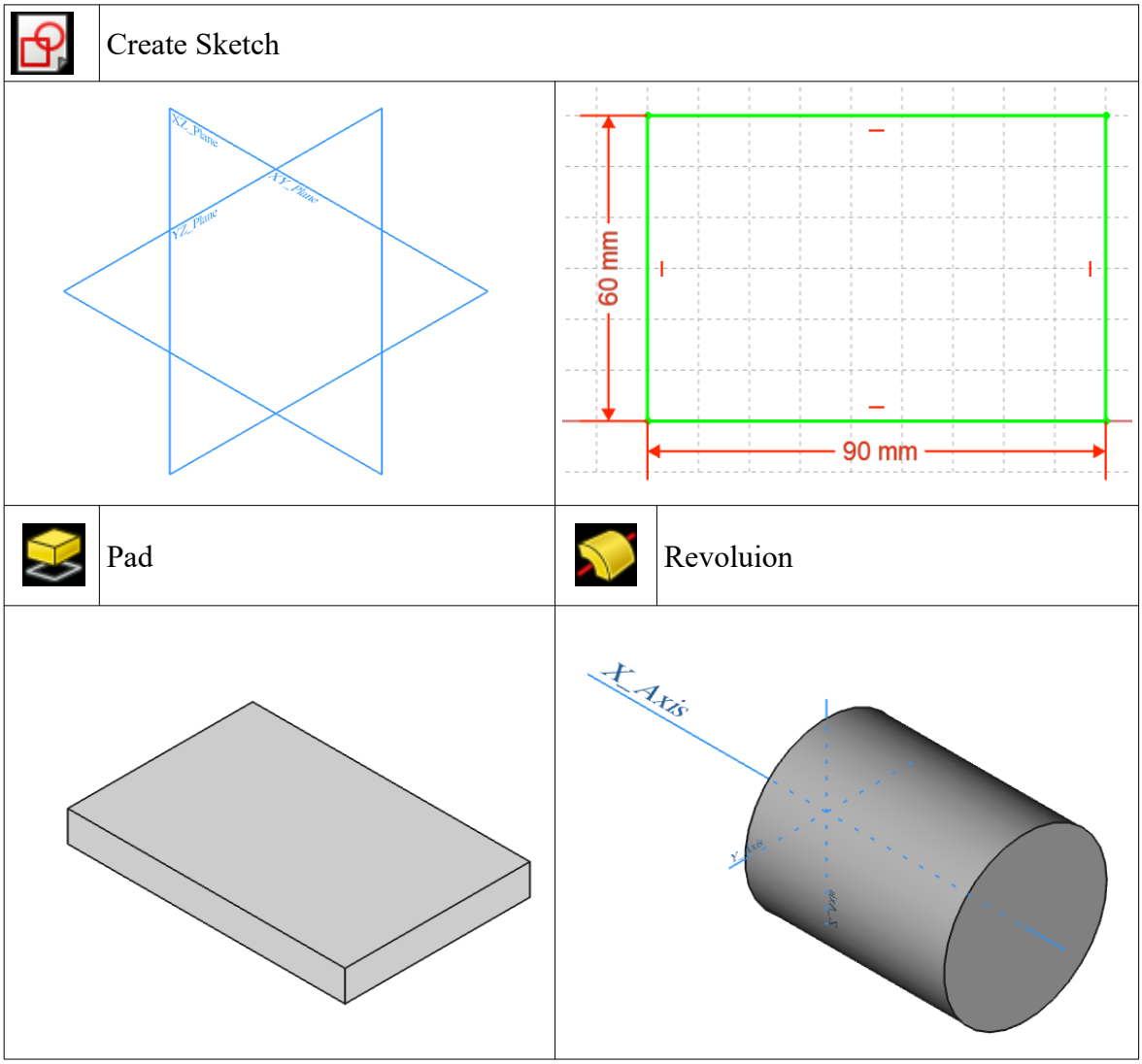
[8] Finite Element Method – Metoda Konečných Prvků – Numerická metoda sloužící k simulaci průběhu napětí,deformací atd.

Na startovní stránce je možné přejít rovnou k vytvoření nového dílu pod rychlým odkazem Tvorba dílu v oblasti Zahájení nového projektu, ostatně jak je vidět na obrázku (Obr. 7). Po přejetí myší na odkaz je zobrazen krátký popis postupu tvorby modelu. Kde je prvním krokem vytvoření skici. Potom následuje její vysunutí do prostoru. Proces se dále v podstatě opakuje kdy je znovu vytvořena skica ale tentokrát je základem jedna z ploch již vysunutého modelu.



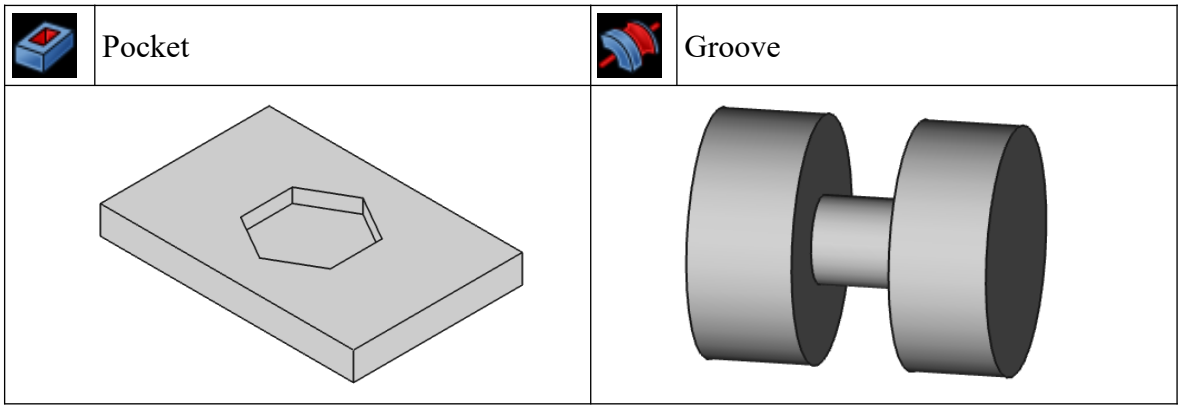
Obr. 7: Rychlý odkaz na tvorbu dílu

Uživatel může buďto zvolit tento rychlý odkaz a nebo vytvořit nový soubor a přepnout do jednoho z modulů pro tvorbu modelu. Na následujícím obrázku (Obr. 8) je vidět zjednodušený postup tvorby modelu v PartDesign modulu. Prvním krokem je vždy zvolení funkce Vytvoření skici *(Create Sketch)* a poté volba roviny souřadného systému, kam se má požadována skica promítnout. V tomto případě je zvolena rovina XY. Při vytváření skici jsou k dispozici základní tvary jako čára, kružnice, oblouk, obdélník ale také třeba mnohoúhelník či ovál. Tyto tvary je dále nutné určitým způsobem omezit a docílit tak požadovaného tvaru. K tomu slouží Vazby náčrtu *(Constraints)*, které omezují vybraný tvar. Existují dva typy vazeb. Jedním jsou vazby se zadáním hodnoty. Touto hodnotou je délka, úhel či průměr zvoleného prvku. Druhý typ vazeb souvisí s geometrickým tvarem a polohou prvku. Sem patří například totožnost bodů, rovnoběžnost či kolmost přímek a další podobné vazby. Na obrázku (Obr. 8) v části tvorby skici je na pravém obrázku vidět zavazbený náčrt, který má vodorovnou vzdálenost mezi dvěma body 90 mm, horizontální vzdálenost 60 mm. Dvě vodorovné úsečky s horizontální vazbou a dvě svislé s vazbou vertikální. Po tvorbě skici je dalším krokem její prostorové vyjádření. To lze provést dvěma způsoby. Prvním je Vysunutí *(Pad)*, což vysouvá skicu ve směru osy kolmé na zvolenou rovinu. Druhý způsob je Rotací *(Revouliton)*, která promítá skicu dle zvolené osy, ovšem osa rotace nesmí být kolmá na rovinu skici. V obou případech musí být skica uzavřený tvar.



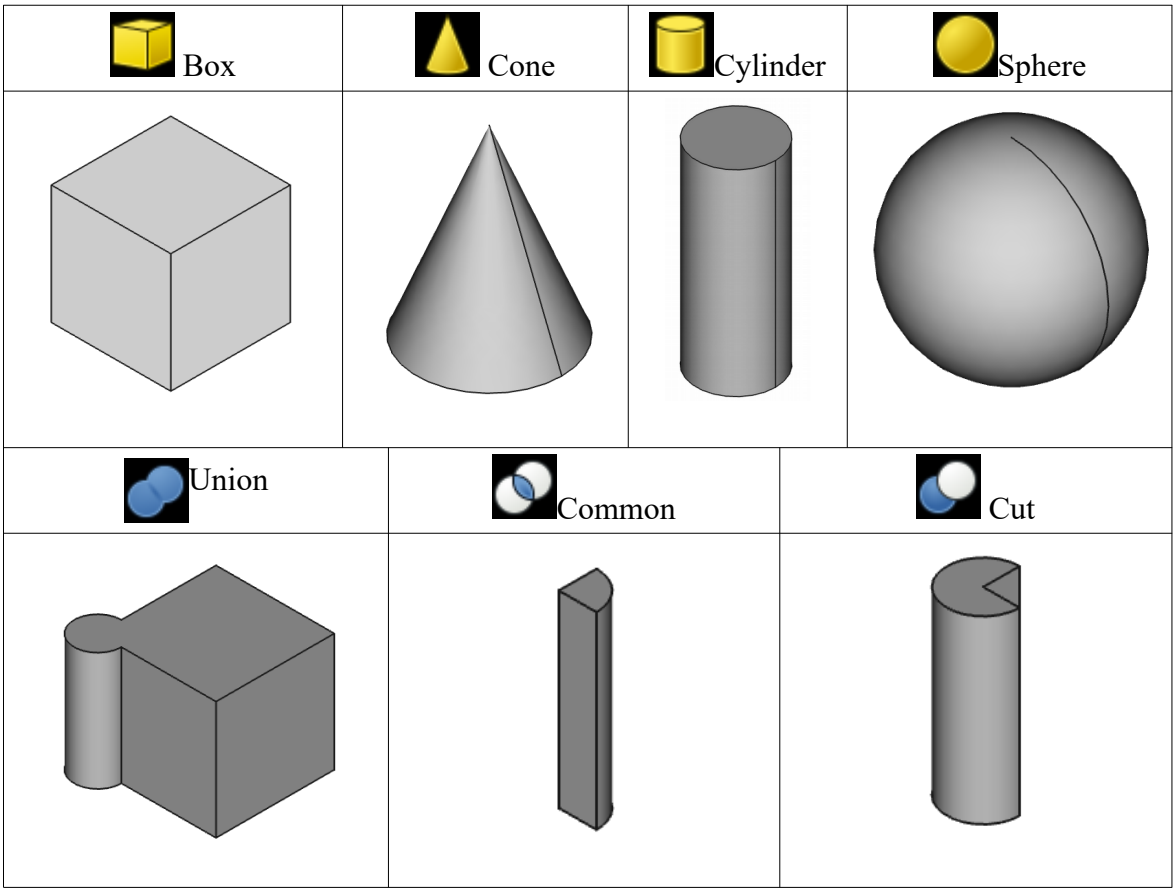
Obr. 8: Tvorba modelu v PartDesign

Pro další úpravu tvaru, k vidění na obrázku (Obr. 9) vlevo je možné použít kapsu *(Pocket)*, která odebírá materiál dle vytvořeného náčrtu a zadané hloubky kapsy. U rotačních součástí je možné použít vybrání *(Groove)* (Obr. 9-vpravo), které vybraný náčrt rotuje kolem zvolené osy a odstraní veškerý materiál nacházející se v této vymezené oblasti. Dále je možné srazit či zaoblit hrany dle přání uživatele. Tento postup je velmi zobecněný a uživatel si tak může vytvořit vlastní postup práce při tvorbě modelu, který mu bude více vyhovovat.



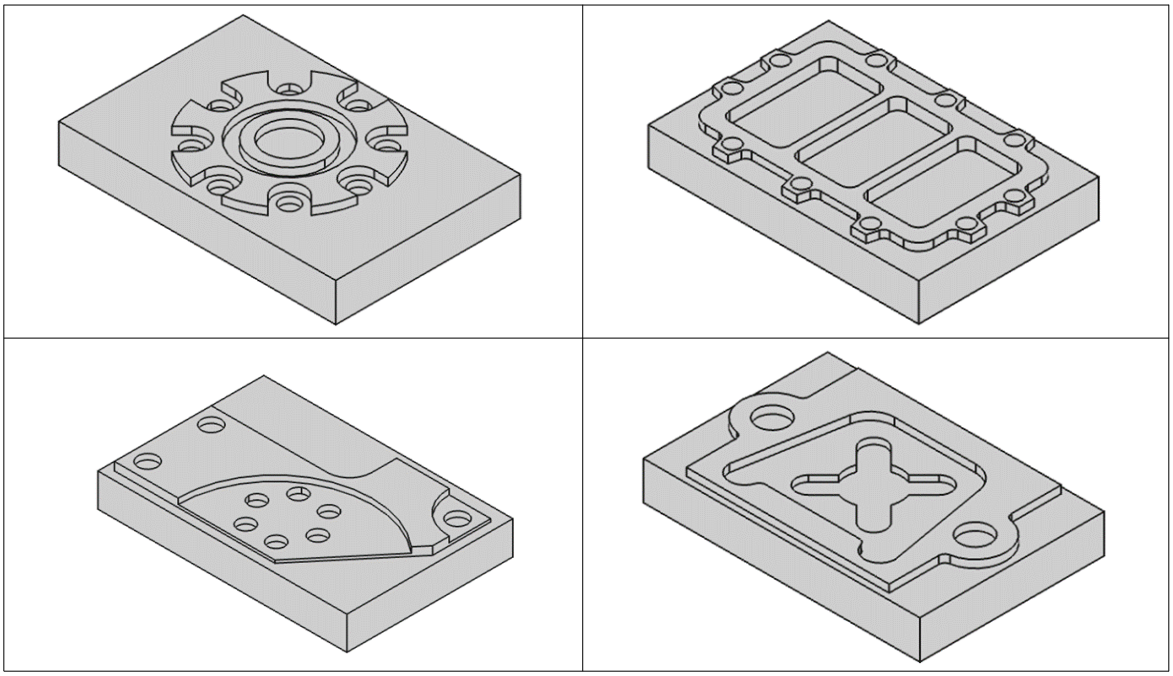
Obr. 9: Kapsa a vybraní

Druhou možností jak vytvořit model je pomocí Part modulu, který vytváří parametrizované modely. Ty specifikuje jako primitivy a umožňuje provádět s těmito modely logické operace. Zmíněné primitivy jsou vidět na obrázku (Obr. 10). Jde o základní tvary jako je krychle *(Box)*, kužel *(Cone)*, válec *(Cylinder)* a koule *(Sphere)*. Všechny parametry těchto prvků je možné upravovat ve vlastnostech zobrazení. Dále je možné s těmito prvky provádět logické operace, jako spojení *(Union)*, průnik společných částí *(Common)* a také vyříznout (*Cut*) část jednoho prvku z druhého. U této funkce záleží na pořadí zvolení jednotlivých prvků. V dolní části obrázku (Obr. 10) je možné vidět logické operace mezi dvěma primitivy, kterými jsou v tomto případě krychle a válec.



Obr. 10: Part modul, primitivy a logické operace

Pro účely této práce byly vytvořeny celkem čtyři modely, jejichž výkresy byly převzaty z bakalářské práce Lukáše Vintra s názvem „Tvorba vzorových příkladů pro výuku NC programování pro nerotační součásti“[1], ostatně jak již bylo zmíněno. Cílem této práce bylo vytvořit cvičné modely pro studenty k výuce NC programování právě pro stroj EMCO Concept MILL 105. Student vytvořil celkem dvacet součástí. Pro ozkoušení funkcí FreeCADu byly vybrány čtyři vzorové součásti, které je možné vidět na obrázku (Obr. 11). Vytvoření těchto modelů proběhlo po důkladném seznámení s modulem PartDesign bez větších obtíží. Přesto je vhodné zmínit že FreeCAD obecně není zatím příliš přívětivý pro nové uživatele a je tak nutné neustále přihlížet do dokumentace. Ta vzhledem k neustálému vývoji software není kompletní a navíc u českého překladu nelze počítat s aktuálností a dostupností veškerých informací. Proto je lepší se držet anglického originálu. Křivka zácviku v tomto software je tak poměrně strmá, což může bohužel nové uživatele odradit od používání.



Obr. 11: Modely vzorových součástí

Oba konstrukční moduly Part i PartDesign jsou svou funkčností rozsáhlejší než je zmíněno výše, ovšem pro účely této práce není nezbytně nutné kompletně popsat veškeré nabízené funkce, ani není jejím cílem vytvářet návod pro používání těchto modulů. Konstrukční moduly FreeCADu lze zhodnotit na dobré úrovni, a tedy se jeví pro běžné konstrukční práce jako dostačující. Je patrné že autoři při vytváření funkcí čerpali z komerčních CAD systémů jako CATIA a SolidWorks, což není v žádném případě naškodu, neboť tyto systémy jsou již vcelku známé a běžní uživatelé jsou na jejich metodiku práce (*workflow*) zvyklí.

3.4.3. Modul pro technologii obrábění

Druhým a pro účely práce mnohem důležitějším rozšířením FreeCADu je modul Path. Ten slouží, jak již bylo zmíněno, ke tvorbě drah nástroje obráběcího procesu. Zatím je používán pouze pro frézování ale v budoucnu je v plánu i soustružení, které ovšem bude mít pravděpodobně samostatný modul pro lepší přehlednost.

K vytvoření drah je nejprve nutné mít k dispozici model, dle kterého se dráhy vytvoří. Uživatel si buďto může vytvořit model ve výše zmíněném Part/PartDesign modulu a nebo importovat 3D model v jednom z řady formátů, které je možné využívat pro přenos dat mezi ostatními CAx systémy a které FreeCAD podporuje. Jsou to STEP, IGES, OBJ, IFC, DAE z prostorových formátu. Dalšími formáty jsou SVG pro vektorovou grafiku, STL pro Reverzní inženýrství a DXF formát pro 2D data.

Při přepnutí do Path modulu je uživateli představena celá řada nových funkcí[9] jejichž přehled je k vidění v následujících tabulkách rozdělených podle typu použití. Za každou tabulkou následuje podrobný popis jednotlivých funkcí.

V tabulce Nastavení projektu (Tab. 3) jsou k vidění operace, které slouží nejen k prvotnímu nastavení obráběcího procesu, ale také i ke kontrole zadaných hodnot a nebo realizaci samotného postprocesingu.

Project Setup – Nastavení projektu	
Název operace	**Stručný popis**
 Job	Vytvoří Job – CNC projekt pod který spadají veškeré další funkce
 Postprocess	Exportuje veškeré operace do ISO kódu stroje dle zvoleného postprocesoru
 Path Errors	Zkontroluje u vybraného Job přítomnost důležitých hodnot
 G-Code Inspector	Zobrazení interního ISO kódu FreeCADu (CL dat) na vybrané operaci nebo celém jobu
 Simulator	Simulace obráběcího procesu

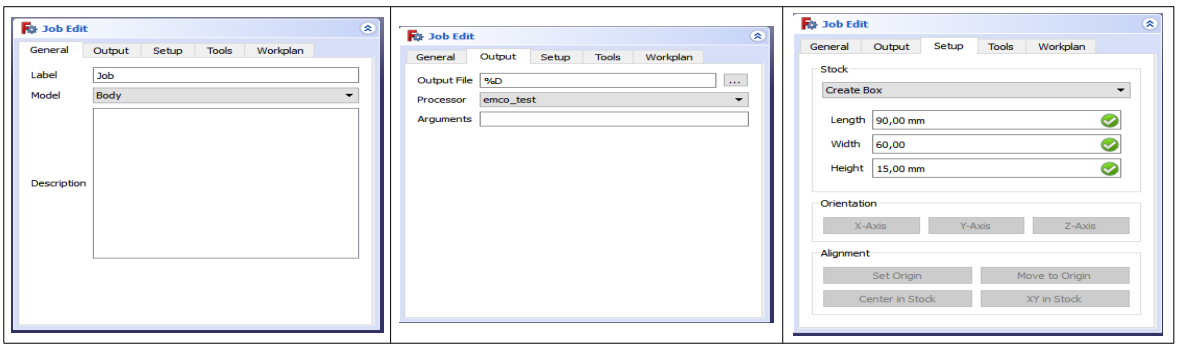
Tab. 3: Project Setup [13]

 **Job**

Tato operace vytvoří nový objekt Job v aktivním dokumentu, kde je možné definovat seznam nástrojů, obrobek a polotovar. Obsahuje několik záložek pro uživatelské nastavení. První tři jsou k vidění na obrázku (Obr. 12), kde pod první záložkou vyobrazenou v levé části obrázku s názvem Obecné *(General)* je pojmenování *(Label)* obráběcího procesu, volba referenčního modelu a libovolný popis objektu *(Description)*. Druhou záložkou uprostřed obrázku je nastavení výstupu *(Output)*. Zde uživatel volí název souboru a cestu kam má být výstupní soubor uložen *(Output File)*. Může si zvolit buďto přímo zvolit cestu k souboru a jeho název. Druhou možností je využít některý z příkazů anebo využít kombinaci několika příkazů například „%D“ uloží výstupní soubor do stejné složky jako je model aktivní součásti. Další příkaz „%d“ pojmenuje výstupní soubor stejně jako je právě otevřený soubor. Po určení výstupu následuje volba postprocesoru *(Postprocessor)* a jeho doplňujících argumentů *(Arguments)* pokud jimi zvolený postprocesor oplývá. Třetí

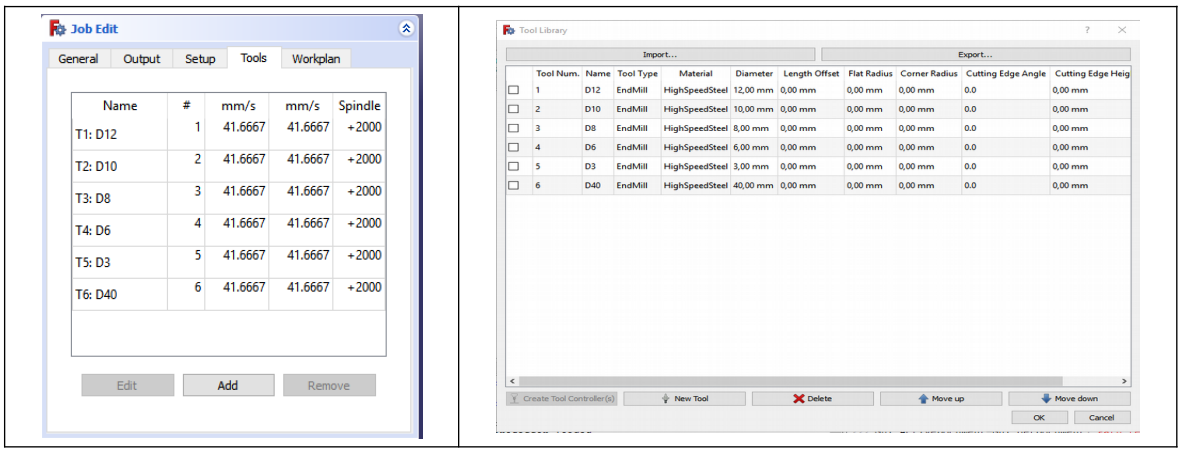
[9] Jde ve skutečnosti o obráběcí strategie. V technologickém postupu by tomuto nejlépe odpovídal operační úsek.

záložkou nacházející se vpravo je nastavení *(Setup)*. To slouží k definici polotovaru, jeho orientaci a zarovnání v prostoru. Také je zde možné nastavit počátek souřadného systému.



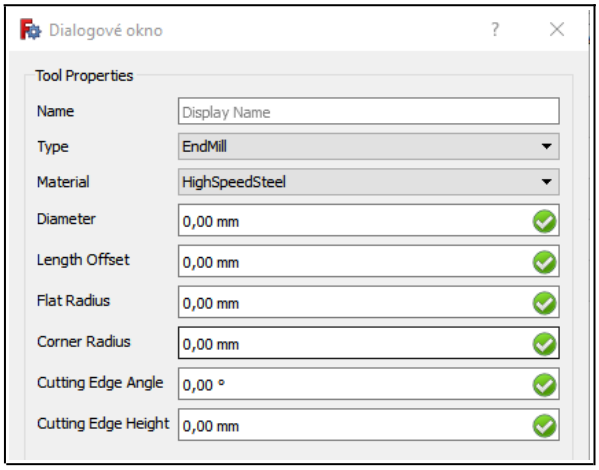
Obr. 12: Job edit – Obecné, Výstup, Nastavení

Čtvrtou záložku Job edit k vidění na obrázku (Obr. 13) vlevo jsou Nástroje *(Tools)*. Zde je možné přidávat, odstranit a upravovat rychlosti jednotlivých nástrojů v projektu a stejně tak otáčky vřetena *(Spindle)*. Poměrně zvláštním zjištěním bylo, že posuv nástroje je zadáván v milimetrech za sekundu (mm/s) namísto standardních v milimetrů za minutu (mm/min). FreeCAD umožňuje zadání v mm/min ale výsledek přepočítává zpět na mm/s. Po kliknutí na tlačítko Přidat *(Add)*, se uživateli otevře nová tabulka s názvem Knihovna nástrojů *(Tool library)*, kde je možné importovat, exportovat seznamy nástrojů, přidávat nové nástroje a měnit jejich pořadí.



Obr. 13: Job edit - Nástroje, Knihovna nástrojů

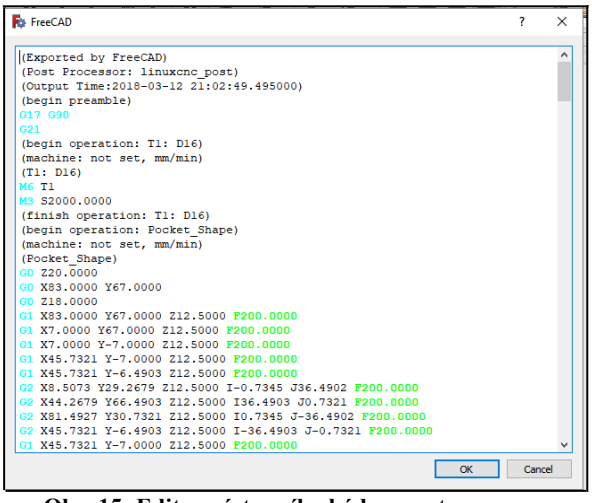
Po přidání nového nástroje se používá tlačítko Nový nástroj *(New Tool)*. Uživateli je představeno dialogové okno, které je vidět na (Obr. 14) kde si uživatel definuje nový nástroj. Parametry které zde uživatel definuje jsou: Jméno nástroje *(Name),* jeho typ *(Type)*, Material, Průměr *(Diameter)*, Délku *(Lenght Offset),* Plošný rádius *(Flat radius),* Úhel řezné hrany *(Cutting Edge Angle) a* Výšku řezné části *(Cutting Edge Height)*. Pro fungující generování drah je nutné zadat minimálně Průměr nástroje. Pokud chce uživatel využít i simulaci obráběcího procesu je nezbytné vyplnit také Výšku řezné části nástroje. Poslední záložka pracovní plán (*Workplan*) slouží k určení a případné změně pořadí jednotlivých obráběcích strategii.



Obr. 14: Vlastnosti nového nástroje

Postprocess

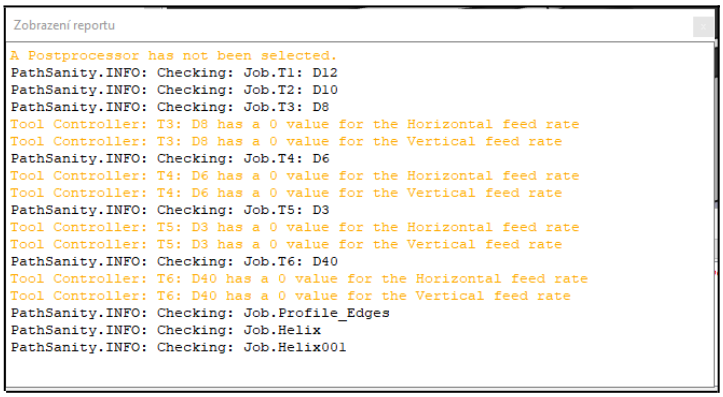
Jak může být z názvu této funkce patrné, její funkce je postproces, neboli převedení vnitřního kódu CAM systému na kód cílového stroje pomocí zvoleného postprocesoru. Pokud uživatel zvolí postprocesor již v nastavení projektu (Job) je pouze vyzván k pojmenování exportovaného souboru. V opačném případě je mu představena tabulka dostupných postprocesorů a poté žádost specifikace cesty uložení. Ještě před uložením souboru je zobrazen editor kódu, kde je možné ještě před uložením souboru kód ručně doupravit obsahuje-li chybné nebo nežádoucí informace. Zobrazení editoru záleží na informaci obsažené v kódu postprocesoru. To tedy znamená, že všechny postprocesory editor zobrazovat nemusí. Na obrázku (Obr. 15) je zobrazen editor kódu před uložením souboru při použití funkce Postprocess se zvoleným postprocesorem linuxcnc.



Obr. 15: Editor výstupního kódu z postprocesoru

Path Errors

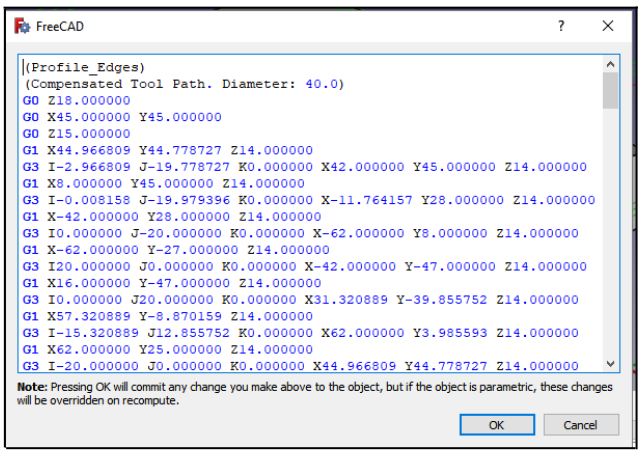
Tato funkce, kontroluje zvolený Job na chyby, tedy hlavně důležité chybějící hodnoty a vypíše do zobrazení reportu výsledek. Na (Obr. 16) je vidět hlášení reportu na Jobu, kde není definován postprocesor a nástroje T3, T4, T5 a T6 nemají definovaný horizontální a vertikální pracovní posuv.



Obr. 16: Hlášení reportu na chyby

G-Code Inspector

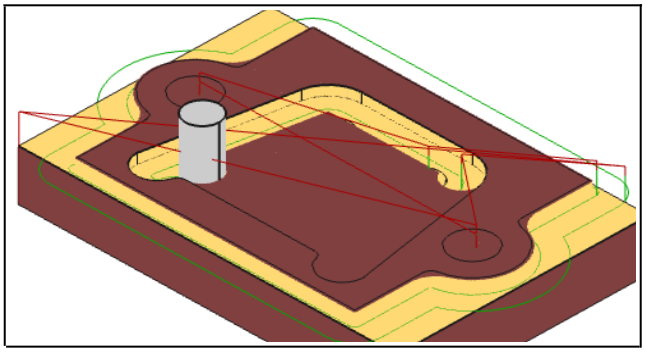
Slouží k zobrazení a kontrole interního „G-kódu“ FreeCADu. Tento již nelze nazývat CL daty, neboť je již částečně převeden do ISO kódu, pro lepší orientaci. Na obrázku (Obr. 17) je vidět ukázka G-Code Inspektoru. Na první pohled se data velmi podobají ISO kódu, ale je zde několik věcí, které je potřeba pro uspokojivý výsledek upravit. Například doplnit technologickou část kódu, přidat čísla řádků či upravit přesnost daných hodnot a tak dále. Ovšem mnohem sofistikovanější způsob je mít postprocesor který tyto nepřesnosti upravuje místo uživatele. Tato funkce se používá především k inspekci drah, díky možnosti zvýraznění právě vybrané části kódu přímo jako zvýrazněné dráhy v pracovním prostoru.



Obr. 17: G-Code Inspcetor

Simulator

Simulátor slouží k vizualizaci definovaných drah pomocí ostatních funkcí Path modulu. Díky tomu je možné detekovat případné chyby před použitím kódu na stroji. Při simulaci je vidět nástroj pohybující se po vytvořených drahách a odebírání materiálu polotovaru. Na obrázku (Obr. 18) je vidět ukázka simulátoru. Hnědou barvou je zobrazen materiál polotovaru a žlutou barvou je míněná obrobená plocha. Občas se stává, že materiál polotovaru zůstává i po průjezdu nástroje. Z důvodu této grafické chyby je doporučeno ještě dráhy ověřit buďto simulací na stroji nebo v nejzazším případě spuštěním programu bez obrobku jak se často také říká „na prázdno“.



Obr. 18: Simulace obrábění

Příkazy pro nástroje *(Tools Commands)* v následující tabulce (Tab. 4) obsahují prozatím jen Správce nástrojů *(Tool Manager)* na který již bylo odkázáno při představení funkce Job viz. (Obr. 13). Uživatel se tedy může do knihovny nástrojů dostat buďto přes Job edit a nebo pomocí Správce nástrojů, pokud si chce nástroje připravit před definicí Jobu.

Tool Comands – Příkazy pro nástroje	
Název operace	**Stručný popis**
 Tool Manager	Správce nástrojů

Tab. 4: Tool Commands [13]

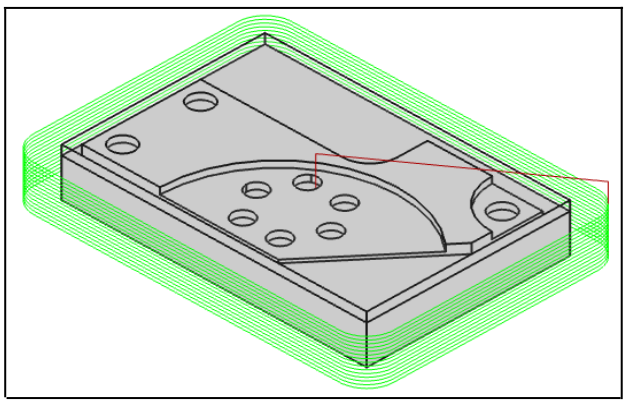
Všechny předešlé operace mají buďto přípravné, pomocné nebo výstupní funkce při tvorbě drah nástroje. Následující seznam operací pojmenovaných Nové operace *(New Operations)* v tabulce (Tab. 5) se tyká především vytváření drah nástroje a definování různých obráběcích strategií.

New Operations – Nové operace	
Název operace	**Stručný popis**
Contour	Vytvoří dráhu podle vnějšího tvaru modelu
Profile from Face	Vytvoří dráhu podle vybrané plochy
Profile from Edges	Vytvoří dráhu nástroje podle vybrané hrany
Pocket	Vytvoří dráhu pro vybranou kapsu
Drilling	Vytvoří vrtací operaci
Engrave	Gravírování
Mill Face	Vytvoří dráhu
Helix	Vytvoří spirálovitou dráhu
3D Pocket	Vytvoří dráhy pro tvarovou kapsu – experimentální funkce

Tab. 5: New Operations [13]

Contour

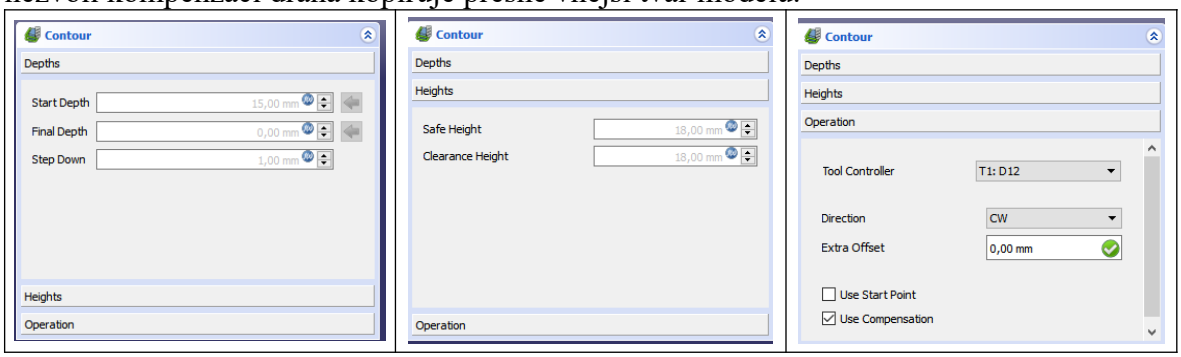
Vytvoří vnější konturu dle definovaného tělesa, jak je možné vidět na obrázku (Obr. 19). Není nutné vybírat žádné plochy ani hrany, FreeCAD automaticky bere zadefinované těleso jakožto podklad pro konturu a z jejího obvodu vytvoří dráhy nástroje.



Obr. 19: Contour

Každá funkce má několik hodnot které je nutné pro její správné fungování nastavit. V případě funkce „Contour“ jde o tři oblasti které je nutné definovat, jak je možné vidět na obrázku (Obr. 20). V levé části obrázku je nastavení hloubek *(Dephts)*, kde se nastavuje počáteční *(Start Depth) a* koncová hloubka (*Final Depth*) spolu s krokem klesání *(Step Down)*. Poté se v druhé záložce nastaví výšky *(Heights)*, kde první hodnotou je bezpečná

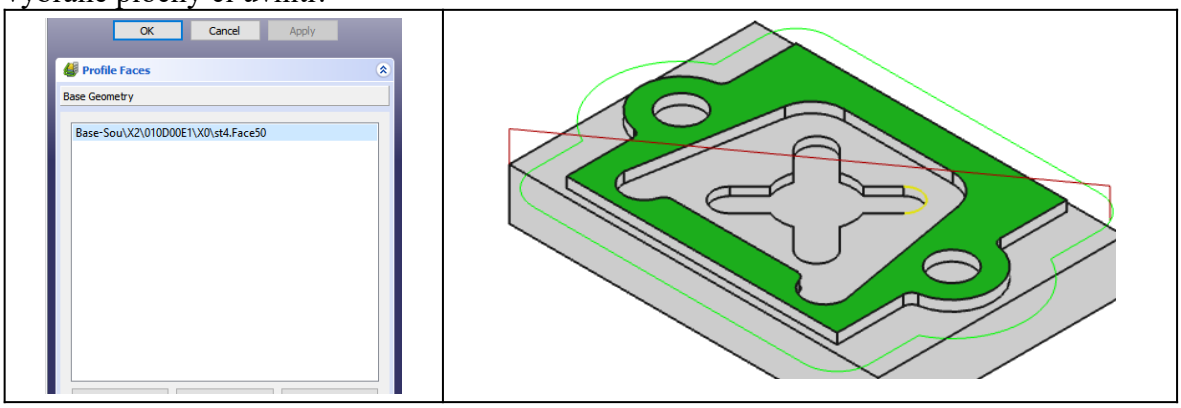
výška *(Safe Height)* kam nástroj najede po výměně rychloposuvem a z této výšky pokračuje pracovním posuvem. Druhý parametr je přejezdová výška *(Clerance Height)* kterou nástroje používají při přejezdech v jednotlivých úsecích. Poslední záložkou nastavení je definice operace *(Operation)*, kde se nastavuje nástroj *(Tool Controller)* a směr posuvu *(Direction)*. Ten může mít dvě hodnoty CW (C*lockWise)* – po směru hodinových ručiček CCW (C*outer-ClockWise)* – proti směru hodinových ručiček. Následujícím parametrem je přídavek *(Extra offset),* který posouvá dráhu o zadaný rozměr od původního rozměru. Funkce použití startovního bodu *(Use Start Point)* definuje počítačení bod operace, jinak je použit vertikální nájezd, což nemusí být v každém případě vhodné. Tento bod je možné upravit ručně ve vlastnostech funkce. Poslední možností je kompenzace průměru nástroje *(Use Compenstaion)*, nebo-li korekce, kde je brán v potaz průměr nástroje a dráha je podle něj posunuta od původního modelu. Pokud uživatel nezvolí kompenzaci dráha kopíruje přesně vnější tvar modelu.



Obr. 20: Nastavení funkce Contour

Profile from Face

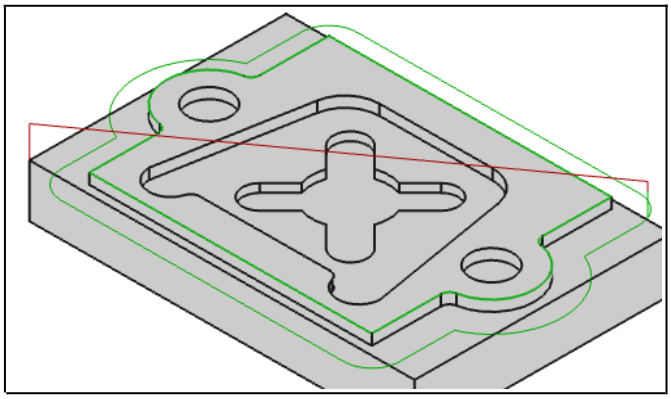
Tato funkce vytváří konturu dle vybrané plochy. Vytvořená kontura může být buďto uvnitř či vně dané plochy. U této funkce je tedy nutné vybrat požadovanou základní geometrii podle které má FreeCAD vytvořit dráhu nástroje. Dále je velmi důležité nastavit parametry operace obdobně jako u předchozí operace „Contour“ s tím rozdílem že, u této operace je nutné definovat základní geometrii *(Base Geometry),* což je vidět v levé části obrázku (Obr. 21). Vybranou geometrií je v daném případě zeleně vyznačená plocha *(Face)* v pravé části obrázku (Obr. 21). U této operace je dále možné zvolit zda se má geometrie tvořit vně vybrané plochy či uvnitř.



Obr. 21: Profile from Face

Profile from Edges

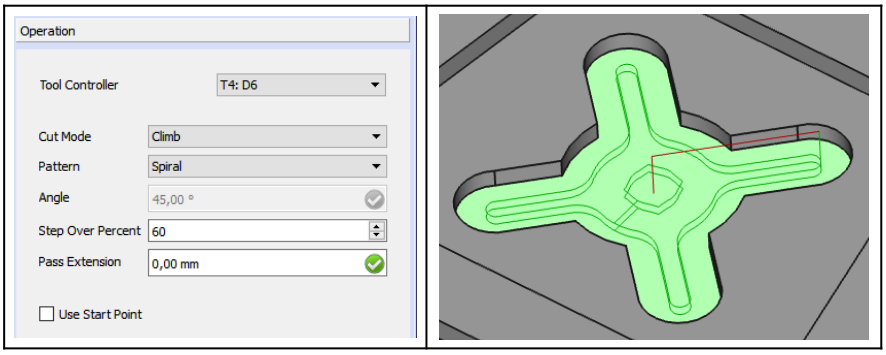
Tato funkce je obdobou předešlé „Profile from Face“, ovšem s tím rozdílem že základní geometrii je v tomto případě okraj či hrana *(Edge)*. Velmi vážnou nedokonalostí této funkce je že zatím nedokáže pracovat s otevřenými konturami, jak by se od takovéto funkce očekávalo. Díky tomu by bylo možné obrábět i vybrané části modelu místo kompletních uzavřených tvarů. Pokud uživatel nezadá do základní geometrie uzavřenou konturu tato funkce nebude fungovat. Na obrázku (Obr. 22) je vidět stejný výsledek jako u „Profile from Face“ (Obr. 21), ovšem v tomto případě je základní geometrií skupina vybraných okrajů, které definují tvar uzavřené konečné kontury.



Obr. 22: Profile from Edge

Pocket

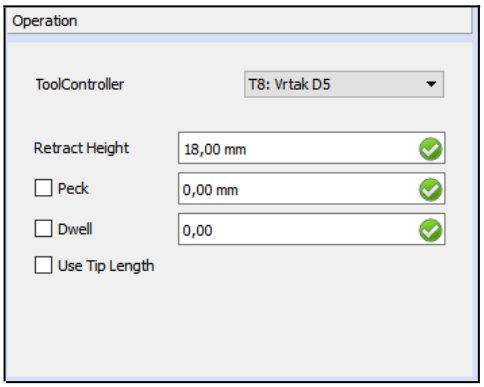
Funkce kapsa *(Pocket)* se používá pro vytváření drah pro vnitřní části modelu (kapsy). V záložce operace u této funkce je k dispozici několik nových parametrů které je možné upravit. Směr řezání *(Cut Mode)*, který může být buďto Climb (CW) nebo Conventional (CCW) obdobně jako směr u funkce „Contour“. Následujícím parametrem je vzor dráhy *(Pattern)*, kterých může být několik: ZigZag, Offset, Spiral, ZigZagOffset, Line, Grid, Triangle. Na obrázku (Obr. 23) je použit vzor Spiral, kdy se nástroj pohybuje po kontuře a přejezdy mezi jednotlivými radiálními dráhami jsou řešeny pracovním přejezdem nástroje ve stejné hloubce, místo výjezdu rychloposuvem do přejezdové výšky a opětovným vertikálním nájezdem nástroje do cílové hloubky na nové souřadnici, jak je to mu řešeno například u vzoru Offset. Dalším parametrem je úhel *(Angle)*. Ten určuje pootočení dráhy vůči souřadnému systému a je možné jej nastavit u vzorů ZigZag, ZigZagOffset, Line, Grid, Triangle. Parametr přesah nástroje *(Step Over Percent)* určuje procentuální překrývání jednotlivých drah nástroje. Hodnota 100 % znamená že se jednotlivé dráhy vůbec nepřekrývají. Parametr *(Pass Extension)* je obdobou přídavku u předešlých funkcí, který posouvá dráhu nastroje od původního tvaru o zadanou hodnotu. V tomto případě je ovšem nutné dbát na průměr nástroje, jinak může dojít k vytvoření neuspokojivé dráhy.



Obr. 23: Pocket

Drilling

Funkce vrtání *(Drilling)* slouží k vytvoření operace vrtání. Tato funkce se využívá pokud se na modelu vyskytuje jedna nebo více děr které je možné místo frézování vrtat. Funkce generuje jeden ze tří cyklů vrtání G81, G82 nebo G83. Cykly se používají k jednoduššímu zápisu pohybu nástroje. Na kartě operací funkce vrtání (Obr. 24) si uživatel zvolí nástroj *(ToolController),* návratovou výšku nástroje *(Retract Height).* Pokud uživatel nezaškrtne další možnosti Path generuje cyklus vrtání G81. Při zaškrtnutí prodlevy *(Dwell)* zůstává nástroj na konečné hloubce po zadanou hodnotu v sekundách a je generován cyklus G82. Pro vrtání hlubších děr je možné použít funkci přerušení *(Peck)*, což generuje cyklus G83. Kdy se vrtá ve vzdálenostech zadaných hodnot a poté se vrací na hodnotu návratové výšky. Posledním parametrem, který si může uživatel zvolit je, že jako výchozí bod bude brána špička vrtáku (*Use Tip Length*).

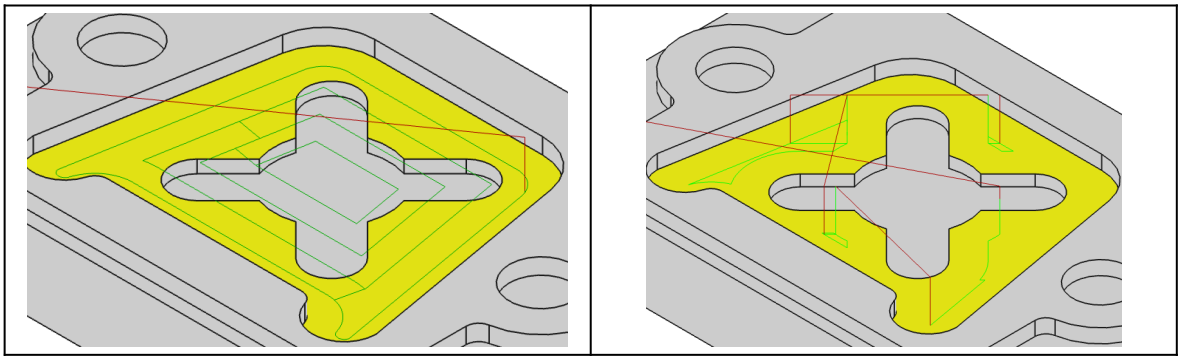


Obr. 24: Záložka operace u funkce vrtání

Mill Face

Funkce frézování plochy *(Mill Face)* vytvoří dráhu dle vybrané plochy. Jde o experimentální funkci přidanou ve verzi 0.17. Při výběru uživatel musí specifikovat plochu *(Face)*, kterou tato funkce použije jako hranice a vytvoří dle ní dráhy. Tvar drah může mít několik typů stejně jako u funkce kapsy. Nastavení dalších parametrů je velmi obdobné jako u ostatních funkcí. Ovšem na rozdíl od kapsy lze tuto funkci použít i na části objektů které obsahují více prvků, jako je tomu na obrázku (Obr. 25). V jeho levé části je vybrána žlutá plocha a funkce „Mill Face“ vytváří dráhy přes celou plochu a ignoruje další

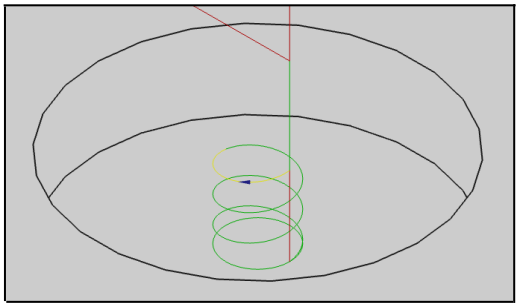
prvek. V pravé části obrázku je na stejnou plochu použita funkce „Pocket“ a je možné vidět že bere v potaz další prvek a jeho okraj bere jako omezení při vytváření drah.



Obr. 25: Mill Face a Pocket

Helix

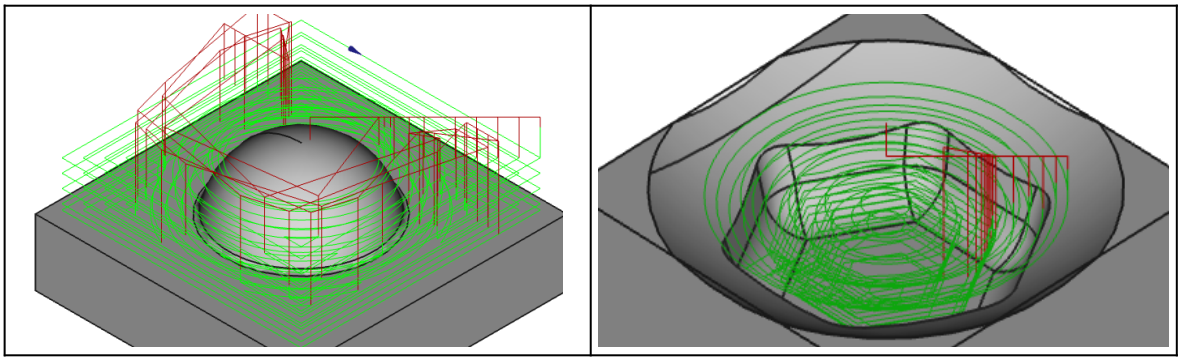
Funkce spirály (*Helix)* vytvoří spirálovou dráhou pro vybranou geometrii, což je možné vidět na obrázku(Obr. 26). Tato funkce je nejvhodnější pro frézování děr, které není možné zhotovit, vzhledem k jejich rozměru nebo tvaru, pomocí vrtání. Nastavení funkce obsahuje obdobné parametry jako předchozí zmíněné funkce.



Obr. 26: Helix

3D Pocket

Funkce slouží pro obrábění tvarových kapes *(3D Pocket)*. Jde o experimentální funkcí. Lze ji použít pro vytvoření drah pro vnější i vnitřní tvarové plochy jak je vidět na obrázku (Obr. 27). Možnosti nastavení této funkce jsou velmi obdobné s funkcí kapsy. Další funkcí pro tvarové plochy je 3D Surface, která by měla sloužit k lepšímu definování drah pro vnější tvarové plochy. Ta je ovšem zatím bohužel ve výchozí instalaci vývojové verze nefunkční, což je neblahá skutečnost, neboť právě vytváření drah pro tvarové plochy je jedním z hlavních důvodů použití CAM systémů. Stejně jako ostatní experimentální funkce i tyto mají buďto nekompletní nebo žádnou dokumentaci.



Obr. 27: 3D Pocket pro vnější a vnitřní tvar

Funkcím úpravy drah *(Path Modification)* v následující tabulce (Tab. 6) není nutné věnovat větší pozornost. Funkce kopie *(Copy)* vytváří parametrickou kopii zvolené dráhy. Tato parametrická kopie je svázána s původní kopii a pokud dojde k upravení původní dráhy změní se i její kopie. Na rozdíl od jednoduché kopie *(Simple Copy)*, která není s původní kopii parametricky svázána. Poslední funkcí patřící do úpravy drah je řada *(Array)*. Ta obdobně jako kopie vytvoří duplikát původní dráhy, ovšem v tomto případě je možné nastavit počet kopii a určit jejich směr.

Path Modification – Úprava drah	
Název operace	**Stručný popis**
Copy	Vytvoří parametrickou kopii vybrané dráhy objektu
Simple Copy	Vytvoří neparamerickou kopii vybráné drahy objektu
Array	Vytvoří pole/řadu z vybrané dráhy

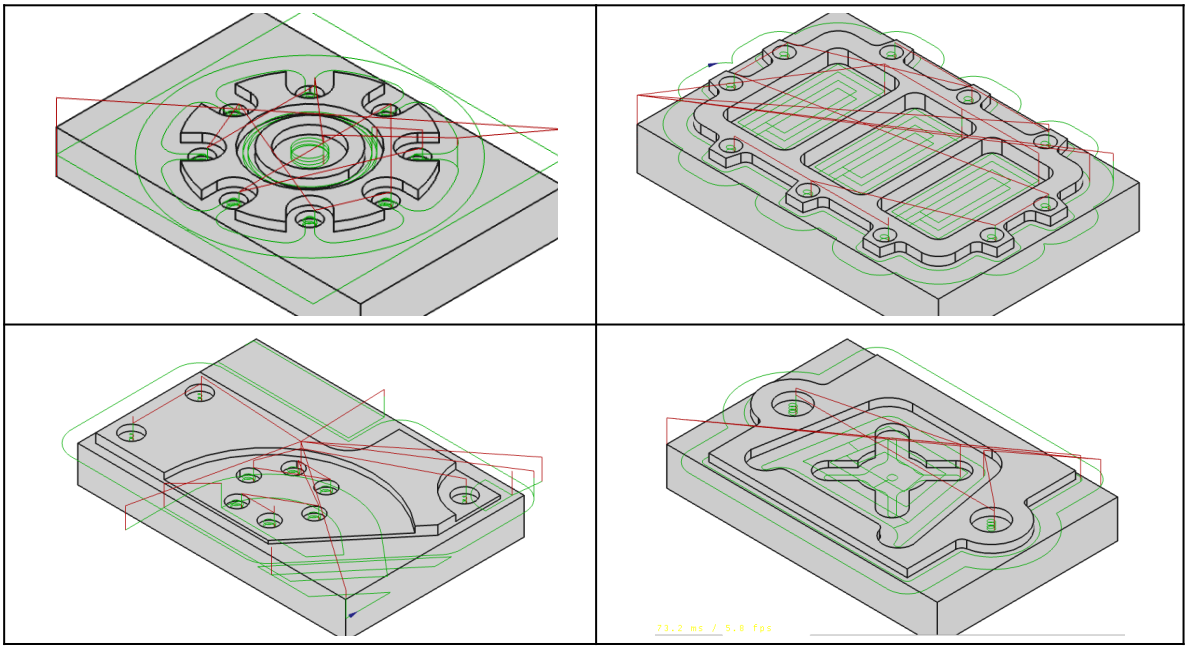
Tab. 6: Path Modification [13]

Poslední skupinou funkcí jsou pomocné nástroje *(Helpful Tools)* které je možné vidět v tabulce (Tab. 7). Patří sem funkce uzavření smyčky *(Complete Loop)*, jejíž funkcí je uzavření smyčky vycházející z vybrané hrany. Tuto funkci je možné využít například při výběru geometrie pro „Profile from Edges“, kde je nutné pro funkční výsledek zadat uzavřenou smyčku tvořenou z hran. Funkce oblasti *(Feature Area)* je experimentální funkcí, která nemá zatím vytvořenou dokumentaci. Při použití vytváří kopii vybrané plochy, ze které je možné dále vytvářet dráhy ale nespadají do stromu operací v Jobu.

Helpful Tools – Užitečné nástroje	
Název operace	**Stručný popis**
Complete Loop	Uzavře smyčku z vybraného okraje
Feature area	Vytvoří oblast z kopie vybrané plochy

Tab. 7: Helpful Tools [13]

Pro zmíněné vzorové součásti v podkapitole 3.4.2, byly vytvořeny dráhy nástroje. Modely spolu s dráhami je možné vidět na následujícím obrázku (Obr. 28). Modul Path není zatím kompletní. Ostatně celý FreeCAD je dosud pouze ve vývojové verzi. Není proto objektivní jej příliš kritizovat nebo vůbec porovnávat s jeho komerčními variantami. Hlavní nevýhodou modulu Path je že zatím není možné vytvářet dráhy pro otevřené kontury a přidávat k dráhám nájezdy a odjezdy. Nicméně lze nastavit alespoň počáteční bod[10] funkce. Další znatelnou nedokonalostí je omezená možnost práce s tvarovými plochami, což bývá jak již bylo zmíněno, jednou ze zásadních předností CAM systému. Přinejmenším nestandardní je fakt, že pracovní posuvy pro jednotlivé nástroje jsou zadávány v milimetrech za sekundu (mm/s) namísto standardních milimetrů za minutu (mm/min). Tyto nedokonalosti software moc nepřispívají k jeho představení novým uživatelům, kteří hledají vhodnou bezplatnou CAM alternativu. Na druhou stranu nabízí zajímavou možnost pro další oblasti vývoje této, už tak zajímavé aplikace. Stále je nutné brát v potaz, že jde o bezplatný a především stále vyvíjený projekt vytvářený skupinou nadšenců a ne o profesionální společnost distribuující CAx řešení pro CNC stroje. V budoucnu lze očekávat odstranění již výše zmíněných nedostatků a doplnění dalších funkcí kterými oplývají ostatní systémy. Díky tomu by se FreeCAD mohl v následujících letech stát zajímavou alternativou komerčním systémům.



Obr. 28: Vytvořené dráhy pro vzorové součásti

3.4.3.1. CL data

Pro tvorbu postprocesoru je nezbytné znát v jaké formě vystupují data z CAM systému. V podstatě jde o interní kód software, který v sobě nese jednak geometrické informace o poloze nástroje a také technologické informace týkající se řezných podmínek. Tomuto kódu se říká CL data[11]. V případě FreeCADu má tento kód tvar podobný ISO kódu, kde G0 reprezentuje rychloposuv *(Rapid) a* G1 pracovní posuv *(Feed)*. Přesto tento kód není

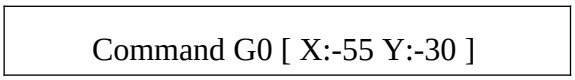
[10] Start point – Startovní bod odkud dráha nástroje začíná
[11] Cutter Location data – informace o poloze řezného nástroje

samostatně použitelný pro správné fungování na cílovém stroji. Vývojáři se pro tento tvar rozhodli kvůli minimalizaci nutnosti překladu interního kódu na ISO formát, což je velmi sofistikované pro tvorbu dalších postprocesorů.

Nejrozsáhlejším zdrojem informací o CL datech FreeCADu je opět i v tomto případě právě jeho dokumentace, která je uváděná v seznamu literatury pod odkazem [11]. Ta bohužel není zatím příliš obsáhlá a informace o CL datech jsou tudíž velmi střídmé. To je poměrně nešťastná skutečnost, neboť jde o klíčovou informaci pro tuto práci. Ovšem FreeCAD je současné době neustále vyvíjen a lze doufat že jeho dokumentace bude v budoucnu doplněna o nezbytné informace.

CL data FreeCADu je možné zobrazit dvěma způsoby. Jedním je použíti postprocesoru s názvem „dumper_post“, který ukazuje surový výstup z Path modulu v editoru G-kódu a nijak jej neupravuje. Nejde tedy o postprocesing ale pouze o zobrazení CL dat. Druhou možností je použít již zmíněnou funkci G-Code Inspector, která ukazuje kód v lehce upravené formě přibližující se více ISO kódu, z důvodu lepší přehlednosti. CL data ke kterým přistupuje FreeCAD jsou ve formátu, který zobrazuje dumper_post.

CL data jsou rozdělena do jednotlivých příkazů *(Commands)*. Každý příkaz je pojmenován buďto G anebo M a je doplněn argumenty v hranatých závorkách. V případě nulové hodnoty příkazu jde o komentář. Na následujícím obrázku (Obr. 29) je vidět příklad příkazu G0, kde jeho argument X má hodnotu 55 a argument Y hodnotu -30.



Obr. 29: Příkaz G0 s argumenty X a Y a jejich hodnotami[11]

Na to aby bylo možné řádně porozumět jednotlivým příkazům, jejichž zástupce je vidět na obrázku (Obr. 29) je vhodné si uvést podporovaných příkazů. Ten je zobrazen v tabulce (Tab. 8). Mezi podporované příkazy se řadí lineární pohyb a to rychloposuvem (G0) a pracovním posuvem (G1). Dalším pohybem je kruhová interpolace po a proti směru hodinových ručiček. Následujícím podporovaným příkazem jsou vrtací cykly G81, G82 a G83. Dále absolutní souřadnice G90 a relativní souřadnice G91. Posledním podporovaným příkazem jsou komentáře v kulatých závorkách.

Příkaz	Popis	Argument
G0 rapid move	Rychloposuv	X,Y,Z,A,B,C
G1 normal (Feed) move	Pracovní posuv	X,Y,Z,A,B,C
G2 ClockWise arc	Oblouk ve směru hodinových ručiček	X,Y,Z,A,B,C,I,J,K
G3 CounterClockWise arc	Oblouk proti směru hodinových ručiček	X,Y,Z,A,B,C,I,J,K
G81, G82, G83 drill	Vratcí cykly	X,Y,Z,R,P,Q
G90 abouslte coordinates	Absoultní souřadnice	-
G91 relative coordinates	Relativní souřadnice	-
(Message)	(Komentář)	-

Tab. 8: Podporované příkazy[11]

Na následujícím obrázku (Obr. 30) je ukázka CL dat FreeCADu získaných pomocí „dumper“ postprocesoru, který generuje interní kód systému beze změny do editoru kódu a neukládá soubor na disk. První část jsou úvodní komentáře v kulatých závorkách, kde je oznámeno že jde o výstup generovaný pomocí dumper postprocesoru. Další informací je že následující data nejsou použitelná pro samotné řízení stroje, jak je na první pohled patrné, ale slouží pouze pro inspekci „syrových“ dat. Uživateli je dále doporučeno aby pro zvolil postprocesor přímo pro svůj cílový stroj. To v tomto případě není možné neboť takovýto postprocesor zatím neexistuje. Po této úvodní části, následují samotné příkazy. Kde jsou již zmíněné příkazy G a M s jejich argumenty, rozdělené po jednotlivých operacích. Před každou operací, která má jiný nástroj než ta předchozí následuje výměna nástroje a to pomocí příkazu M06 s argumentem T.

```
(This output produced with the dump post processor) (Úvodní komentáře)
(Dump is useful for inspecting the raw commands in your paths)
(but is not useful for driving machines.)
(Consider setting a default postprocessor in your project or )
(exporting your paths using a specific post that matches your machine)
(Path: T1: D16) Výměna nástroje
Command (T1: D16) [ ] Prázdný příkaz
Command M6 [ T:1 ] Příkaz M6 s argumentem T jehož hodnota je 1
Command M3 [ S:2000 ]
(Path: Pocket_Shape) Kapsa
Command (Pocket_Shape) [ ]
Command G0 [ Z:20 ] Rychlopostuv na souřadnici Z
Command G0 [ X:83 Y:67 ]
Command G0 [ Z:18 ]
Command G1 [ F:3.3333 X:83 Y:67 Z:12.5 ] Pracovní posuv na souřadnice XYZ posuvem F
Command G1 [ F:3.3333 X:7 Y:67 Z:12.5 ]
…
Command G2 [ F:3.3333 I:-0.73447 J:36.49 K:0 X:8.5073 Y:29.268 Z:12.5 ]
Kruhová interpolace
```

Obr. 30: Ukázka CL dat FreeCADu

Postprocesor tak bude mít za úkol převést výše zobrazené data (Obr. 30) na formát ISO kódu. Příklad tohoto překladu je znázorněn v následující tabulce (Tab. 9). Jde tedy o odstranění řetězce „Command“ dále o převedení informací v hranatých závorkách a odstranění dvojtečky u hodnot argumentu. Další nesrovnalostí je že všechny hodnoty které nejsou zadány jako celé číslo, mají čtyři desetinná místa za tečkou, což bude nutné zmenšit na tři a v případě posuvů je převést na celé číslo. Především je nutné přidat

informace, které v CL datech nejsou obsaženy jako například volba pracovní roviny, vypnutí korekce, definice nulového bodu atd.

Vstupní CL data	Výstupní ISO formát
`Command G1 [ F:3.3333 X:83 Y:67 Z:12.5 ]`	`G1 X83 Y67 Z12.5 F3.33`

Tab. 9: Ukázka převodu CL dat na ISO kód

Ukázku vybraných úseků CL dat ze systému CATIA je možné vidět na obrázku (Obr. 31). První informací je zápis komentářů, které jsou v tomto případě značeny symbolem $. Komentovány jsou většinou názvy operací jejich začátek a konec. Výměna nástroje probíhá pomocí příkazu LOADTL *(Load Tool)* za nímž následují příkazy pro posuv FEDRAT (*Feedrate*) s hodnotou v milimetrech za minutu *(MMPM – milimeters per minute*) a příkaz pro otáčky vřetena SPINDL (S*pindle*) v otáčkách za minutu (*RPM – Revolutions per minute*). Interpolace lineárních pohybů je definována příkazem GOTO (jdi na) jemuž předchází buďto FEDRAT (pracovní posuv) s zadanou hodnotou v MMPM nebo RAPID (rychloposuv). Kruhová interpolace je specifikována TLON, GOFWD/ (CIRCLE/)

```
$$ OPERATION NAME : Profile Contouring.2 Název operace
$$  Start generation of : Profile Contouring.2
LOADTL/3,1 Nástroj
FEDRAT/  300.0000,MMPM Posuv
SPINDL/   70.0000,RPM,CLW Vřeteno
...
FEDRAT/ 1000.0000,MMPM Posuv
GOTO  /  -62.00000,  -48.00000,   14.00000 Souřadnice posuvu
RAPID Rychloposuv
GOTO  /  -63.00000,  -47.00000,   14.00000
...
INDIRV/    0.00000,    1.00000,    0.00000 Vektor směru pohybu
TLON,GOFWD/(CIRCLE/-42.00000,8.00000,14.00000,$21.00000),
ON,(LINE/-42.00000, 8.00000, 14.00000,$-42.00000, 29.00000,14.00000)
Kruhová interpolace
```

Obr. 31: Ukázka CL dat ze systému CATIA

Při porovnání CL dat mezi FreeCADem a CATII je na první pohled patrný rozdíl. FreeCAD má mnohem blíže ISO kódu než CATIA, což je pozitivní pro tvorbu postprocesorů neboť odpadá nutnost překladu dalších hodnot, ostatně jak již bylo zmíněno. Autoři vytvářeli tento výstup s tím předpokladem že postprocesory budou vytvářet samotní uživatelé FreeCADu.

3.5. Python

Předem je vhodné zmínit že v této kapitole je vysvětlen základní princip Pythonu a jeho obecné fungování, ale v žádném případě nejde o podrobný návod a popis jeho veškerých možností. Tento programovací jazyk figuruje v práci jako druhý činitel, neboť je použit pro tvorbu postprocesoru. Jde o open-source multi-platformní software, který je vcelku jednoduchý pro používání a je možné se poměrně rychle naučit jej používat. Jedná se o interpretovaný[12] jazyk, což znamená že na rozdíl od kompilovaných[13] jazyků, jako je například C, je možné program v Pythonu okamžitě spustit bez nutnosti kompilace. Dále je vhodné zmínit že jde o multiparadigmatický[14] jazyk, což tedy znamená že k vytváření kódu je možné přistupovat více způsoby a docílit stejného výsledku. Volba programovacího paradigmatu tak záleží na programátorovi dle jeho preferencí a vhodnosti samotného paradigmatu pro právě vytvářený kód.

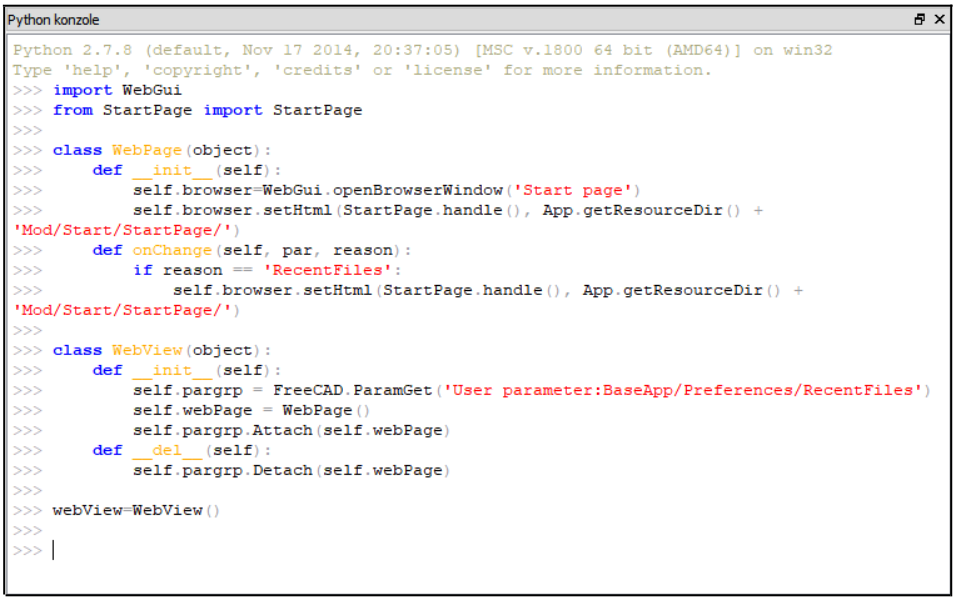
Mezi paradigma které Python podporuje, patří objektově orientované, kde je stěžejním faktorem takzvaný objekt, což může být malá část kódu fungující samostatně a v kódu může být pak použita i vícekrát. Objekt obsahuje informace o svém stavu a ten pak může na příkaz měnit. [14]

Dalším paradigmatem je procedurální, kde je kód sestaven v jasné posloupnosti a jednotlivé příkazy sou vykonávány v postupné sekvenci. Toto paradigma je také považováno za klasické neboť jde o jedno z nejstarších paradigmat programování. Objektové a procedurální paradigma jsou v Pythonu plně podporovány. [14]

Posledním částečně podporovaným paradigmatem Pythonu je funkcionální. Funkce zde bývají aplikovány na výsledky jiných funkcí, namísto přiřazování hodnot k definovaným proměnným jako u ostatních paradigmat. Po procedurálním paradigmatu se jedná o druhé nejstarší paradigma. [14]

Python je možné využít na velké množství věcí od jednoduchých skriptů až po složité programy. Jednou z jeho velkých předností je možnost jej zakomponovat do dalších aplikací, což je právě i případ FreeCADu. Zde je interpretem (překladačem) Python konzole, k vidění na obrázku (Obr. 32) a pomocí které, je možné FreeCAD i ovládat. Jde například o provádění úkonů pro které dosud není k dispozici funkce v grafickém rozhraní, některého z dostupných modulů. Tato konzole zobrazuje veškeré úkony provedené ve FreeCADu jako python kód. Na obrázku je vidět kód při spuštění programu. Znak „>>>“ naznačuje že jde o interaktivní překladač, který po zadání ihned vrací výsledky. Například je možné jej použít jako kalkulačku.

[12] Překlad do strojového jazyka probíhá souběžně s během programu, je vyžadován interpret (překladač), což je software, který program spustí [14]
[13] Program je nejprve přeložen do strojového jazyka (zkompilován) a až poté je možné jej spustit[14]
[14] Programovací paradigma – programovací styl – způsob tvorby kódu



Obr. 32: Python konzole

FreeCAD momentálně využívá Python verze 2.7.8 (2. X) jak je také vidět na obrázku (Obr. 32), ovšem Python je k dispozici ve dvou verzích a to 2. X a 3. X. Rozdíl mezi nimi je ve stylu zapisování syntaxe a způsobu vyhodnocování některých vstupů. Další rozdíl je také v tom že verze 3. X je dále vyvíjena a vylepšována, zatímco verze 2. X je ve své konečné fázi s verzí 2.7. V následující tabulce (Tab. 10) je ukázka jen některých změn které se týkají této práce. Například příkaz print v Pythonu 2. X nepoužívá závorky neboť jde o tvrzení, ovšem ve 3. X je print již funkce a proto je nutné její řetězce psát v kulatých závorkách a je dále možné využít argumenty této funkce. Příkaz zde figuruje také kvůli budoucí zpětné kompatibilitě. Další změnou je náhrada znaku není rovno „<>“ za „!=“. To je z důvodu sjednocení obecné syntaxe s ostatními programovacími jazyky, kde operátorem nerovnosti je také znak „!=“. Vstup (*input*) je ve verzi 2. X vyhodnocován, tedy například při zadání vstupu 2*6 je vrácená hodnota 12. Ve verzi 3. X je vrácena řetězcová hodnota „2*6“. Pro výsledek stejný jako v 2. X je nutné před input přidat funkci „eval“ a výsledný příkaz by vypadal takto: eval (input(„2*6“).

Rozdíl	Python 2.X	Python 3.X
print	print „Hello World“	print („Hello World“)
Není rovno	<>	!=
input	raw_imput(„vstupni hodnoty“)	Input („ řetezcová hodnota“)

Tab. 10: Python 2.X/3.X

Pro vytváření složitějších kódu, jejichž podstatu nelze obsáhnout do jediného příkazu na osamoceném řádku postačí základní textový editor. Tím je například v operačním systému Windows aplikace Poznámkový blok. Ovšem doporučená a mnohem sofistikovanější metoda je použití některého z editorů neformátovaného textu.

Soubor se zdrojovým kódem je v podstatě sled znaků, tedy bitů které přesně odpovídají zaznamenaným znakům v ASCII, UTF-8 či jiném kódování. Zatímco soubory formátovaného textu, kromě toho obsahují ještě formátovací znaky. Ty udržují informaci

o tom, že příslušná část textu má charakter nadpisu, že je pro nadpis zvolený specifický font s definovanou velikostí písma. Dále jestli je text kurzívou, tučně či jaká je jeho barva a tak dále.

Mezi aplikace patřící do skupiny editorů neformátovaného textu, které zobrazují text bez formátovacích značek a jsou proto vhodné právě pro tvorbu zdrojového kódu v různých programovacích jazycích, patří například Notepad++. Tato aplikace se vyznačuje, tím že umožňuje skrýt části kódu, které jsou podřadné předcházejícímu tvrzení a programátor tuto část pro další tvorbu nepotřebuje. Například u funkce if po schování je vidět pouze definice podmínky na prvním řádku. Ovšem i přes tuto zajímavou vlastnost je pro Python doporučován IDLE. Tento akronym znamená *Integrated DeveLopment Eviroment*, což je Integrované Vývojové Prostředí. Jeho nespornou výhodou je, že může kód v Pythonu spouštět a ihned tak ověřit jeho správnost.

Další významnou výhodou těchto editorů je zvýrazňování syntaxe (*Syntax highlighting*). Nejedná se o informaci, která se udržuje uvnitř souboru ale editor sám rozpozná určitá klíčová slova (řetězce) a zvýrazní je, ale informaci do souboru nezapisuje. Tato informace je při vytváření programu velmi užitečná neboť zlepšuje orientaci v kódu. Pro účely této práce je použit IDLE, právě díky možnosti okamžitého otestování správnosti kódu.

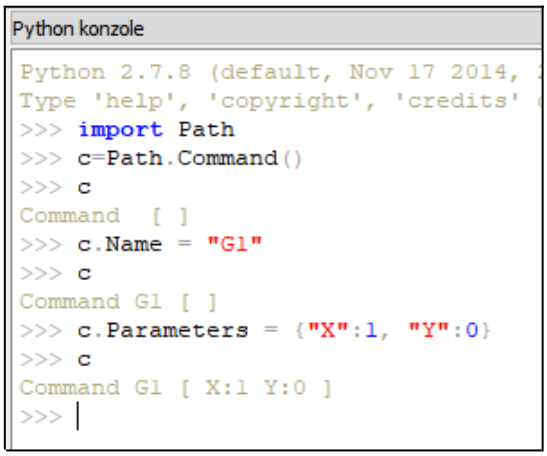
Python disponuje celou řadou vestavěných (*built-in*) funkcí pro práci s hodnotami, proměnnými a textovými řetězci, ostatně jako většina programovacích jazyků. Nicméně za zmínku stojí spíše fakt, že si uživatel může vytvořit vlastní funkce a to pomocí příkazu **def**, což je zkratka pro defined (*definované*). Syntaxe této funkce je znázorněna na následujícím obrázku (Obr. 33). Za příkazem *def* následuje jednoslovný název, případně oddělený podtržítkem, doprovázený vstupními parametry v kulatých závorkách. Dvojtečka za kulatou závorkou charakterizuje začátek bloku funkce. Dalším parametrem je popis či dokumentace funkce, který je sice nepovinný, ovšem z hlediska přehlednosti kódu je doporučeno jednotlivé funkce alespoň okrajově popsat. Následujícím parametrem je již samotná definice funkce, jejíž rozsah záleží na složitosti vytvářené funkce. Příkaz *Return* signalizuje ukončení definované funkce s zadaným výstupním výrazem, ovšem pokud není výraz definován funkce nevrací nic, stejně pokud by byla jako výraz zadána hodnota *None*. [15]

```
def název_funkce(parametry):
    "Popis funkce"
    Definice funkce
    Return [výraz]
```

Obr. 33: Syntaxe uživatelem definované funkce[15]

Ve FreeCADu je již zmíněný interní kód (CL data) reprezentován jako Příkazový objekt (*Command object*). Ten obsahuje tři atributy Jméno (*Name*), Parametery (*Parameters) a* umístění (*Placement)*. Již zmíněné příkazy (*Commands*) jsou definovány pomocí atributu Jméno jehož hodnoty mohou být G0, G1, M3, M6 a komentář. Atribut parametr je doplňující informací příkazu a má následující hodnoty: (X, Y, Z, A, B, C, I, J, K, F, S, T, Q, R, L, H). Poslední parametr umístění slouží pro grafické rozhraní a není tedy pro postprocesor důležitou informací.[11]

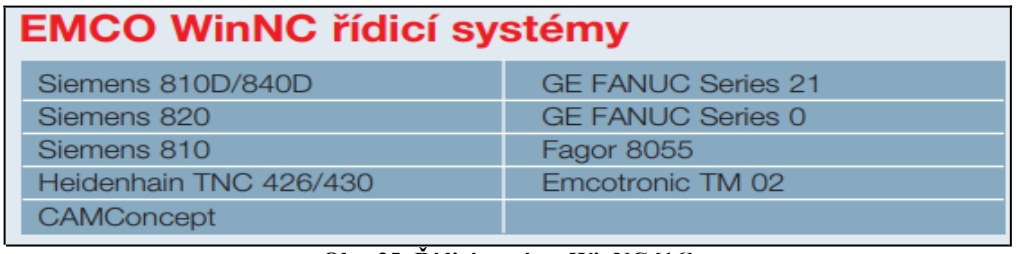
Na obrázku (Obr. 34) je vidět vytvoření objektu Path modulu který obsahuje příkaz G1 s hodnotami X1 a Y0 přímo ve FreeCADu pomocí python konzole. Nejprve je zadefinována hodnota c jako objekt Path modulu (*Path. Command*). Dále je mu přiděleno jméno (c. Name) G1 a poté parametry (c. Parameters) s hodnotami. Objekt v Path modulu je v pythonu reprezentován jako Command([Name], [Parameters], tedy Příkaz ([Jméno],]Parametry]).



Obr. 34: Vytvoření objektu

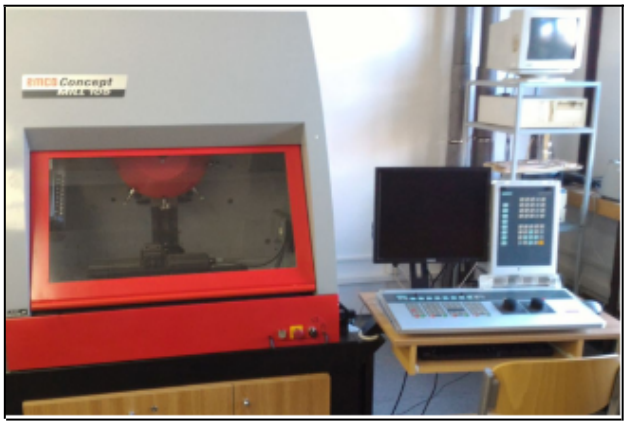
3.6. Frézka EMCO Concept MILL 105

Třetím činitelem je školní stroj EMCO Concept MILL 105, k vidění na obrázku (Obr. 36). Jde o tříosou frézku s revolverovým zásobníkem o deseti pozicích. Tento stroj slouží především pro účely výuky programování NC strojů na KTO. Díky software WinNC a vyměnitelnému ovládacímu panelu je možné pro tento stroj využít několik řídících systémů dle zdroje [16]. Jejich výčet je k vidění na obrázku (Obr. 35) [16].



Obr. 35: Řídicí systémy WinNC [16]

Momentálním řídícím systémem je emulovaný Sinumerik840D přes operační systém Windows pomocí software WinNC[16]. Ovšem v budoucnu dojde k nahrazení tohoto emulátoru za plnohodnotný řídicí systém Sinumerik 828D. Jde o řídící systém od firmy Siemens pro standardní frézky, soustruhy a brusky.



Obr. 36: Výuková frézka EMCO Concept MILL 105 [17]

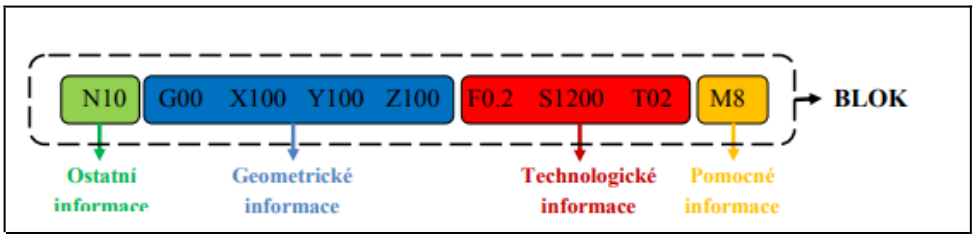
V tabulce (Tab. 11) je možné vidět rozměrové parametry stroje, velikost a maximální zatížení stolu, dále výkon vřetena a nakonec informace týkající se nástrojového systému. Z těchto informací je patrné, že jde o poměrně malý stroj s tomu odpovídajícím výkonem. To je ovšem pro stroj, který se používá zejména pro účely výuky dostačující. Nejvíce se s tímto strojem setkali studenti profesního bakalářského oboru programování NC strojů, kde bylo podmínkou absolvování jednoho z předmětů výroba zadané součásti přímo na tomto stroji.

Parametry stroje	
Rozměry stroje (ŠxDxV) [mm]	1135x1100x1100
Rozměry posuvu stroje X/Y/Z [mm]	200/150/250
Min.vzdálenost čela vřetena od stolu [mm]	95
Max.vzdálenost čela vřetena od stolu [mm]	245
Stůl	
Délka x Šířka [mm]	420 x 125
Maximální zatížení [kg]	10
Vřeteno	
Rozsah otáček [ot.]	150 -5000
Výkon [kW]	1,1
Kroutící moment [Nm]	4,2
Nástrojový systém	
Počet nástrojových pozic [-]	10
Max. průměr nástroje [mm]	55
Max.délka nástroje [mm]	50

Tab. 11: Parametry stroje [16][17]

3.6.1. ISO kód

Již zmíněným konečným výstupem z postprocesoru CAM systému je myšlen specifický kód stroje (například ISO kód), který v sobě nese informace nezbytné pro realizaci výroby požadované součásti. Tomuto kódu se taky často říká NC-kód a informace, které obsahuje jsou k vidění na obrázku (Obr. 37). Jde o technologické, geometrické, pomocné a ostatní informace. Technologické informace se týkají označení nástroje, jeho korekce a především řezných podmínek, kam patří řezná rychlost, posuv a hloubka řezu. Geometrické informace se týkají polohy a tvaru dráhy nástroje. V podstatě určují zda jde o přímkovou nebo kruhovou interpolaci. Pomocné informace obsahují pokyny pro doplňující příkazy, ale i ty jsou nezbytné pro správný chod programu. Patří sem například roztočení vřetene, zapnutí/vypnutí řezné kapaliny nebo ukončení programu. Poslední částí jsou ostatní informace které obsahuji číslo bloků jednotlivých řádků a případně poznámky programátora k jednotlivým blokům. [18]



Obr. 37: Blok NC programu[18]

Variací ISO kódu je celá řada. Každý výrobce má svůj ISO kód specifický. Například podle toho kolik informací může být obsaženo v jednom bloku. Mezi rozšířené výrobce patří právě Siemens se svým systémem Sinumerik, dále FANUC, Heidenhain či Mazak. Existuje dokonce standard formátu bloku v podobě normy ČSN ISO 6983-1 s názvem „*Číslicové řízení strojů. Formát programu a definice adres. Část 1: Formát dat pro polohovací, pravoúhlé a souvislé řídicí systémy*“, ovšem ta byla zrušena bez náhrady. Její učinnost byla od data 10/1992 až do 10/2016. [19]

Formát bloku má tedy u každého řídicího systému svá specifika, ale všechny vycházejí ze společného základu. Tím je míněno, že přípravné funkce jsou ve většině případů značeny G a pomocné M. Ovšem význam jednotlivých adres deklaruje už sám výrobce. V případě Sinumeriku jsou G funkce rozděleny do jednotlivých skupin. Těchto skupin je celkem 62. Blok nemůže obsahovat více než jeden příkaz z dané skupiny. Na obrázku (Obr. 38) je ukázka příkazů spadajících do skupiny šest, což je skupina kam patří příkazy pro volbu roviny.

G group 6: Plane selection						
G command	No. 1)	Meaning	MD20150 2)	W 3)	STD 4)	
					SAG	MM
G17	1	Plane selection 1st – 2nd geometry axis	+	m	x	
G18	2	Plane selection 3rd – 1st geometry axis	+	m		
G19	3	Plane selection 2nd – 3rd geometry axis	+	m		

Obr. 38: G skupina č.6

4. Návrh postprocesoru

Stěžejním bodem práce je, jak může být z jejího názvu patrné, tvorba postprocesoru. Následující kapitola je koncipována tak, že jsou ukázány jednotlivé části kódu postprocesoru, důkladně popsány a vysvětleny. Kompletní kód je možné vidět v příloze (*Příloha č.1*) a na přiloženém CD včetně komentářů. Pro otevření výstupního souboru s příponou „.py“ postačí obyčejný textový editor (Poznámkový blok), ovšem doporučuje se otevírat soubor v některém z již výše zmíněných editorů neformátovaného textu používaných pro vytváření zdrojového kódu, které zvýrazňují syntaxi pythonu.

První ukázka části kódu postprocesoru je k vidění na obrázku (Obr. 39) a obsahuje pouze komentáře informativního charakteru. Symbol „#“ značí že jde o komentář. Prvním z komentářů musí v tomto případě být „#encoding: utf -8“ z toho důvodu, že FreeCAD momentálně používá Python s verzí 2.7 jehož kódování je ASCII. Naproti tomu Python verze 3 má již výchozí kódování UTF-8 a podporuje tak mnohem více znaků. Kam patří například symboly s diakritikou (ě, š, č, ř, ž, ý, á, í, é atd.). Další informace jsou o druhu postprocesoru a důvodu jeho tvorby. Dále je také zmíněno že postprocesorem ze kterého čerpá nejvíce informací je linuxCNC, ostatně jako většina dostupných postprocesorů, jak již bylo zmíněno výše.

```
#encoding: utf -8
#Postprocesor pro EMCO CONCEPT MILL 105
#Vytvoren v rámci DP: "Postprocesor pro zvoleny stroj k SW FreeCAD"
#vychazi z linuxCNC_post.py
```

Obr. 39: Úvodní část kódu

Následující část kódu zobrazená na obrázku (Obr. 40) je import modulů, obsahujících třídy a funkce potřebné pro samotnou funkci programu. Jde o import částí FreeCADu a pythonu pro jejich budoucí využití v samotném programu. První tvrzení „from __ future__ import print_function““, zde figuruje především z toho důvodu, že v současné verzi Pythonu 2.7.8 ve FreeCADu není příkaz *print* definován jako funkce ale pouze jako tvrzení. Tento příkaz dělá z *print* funkci, jak je tomu již v novější verzi Pythonu 3.5.2. Další je import FreeCADu, Path modulu a z FreeCADu navíc ještě jeho systém jednotek. Následujícím importovaným modulem je shelx, který slouží k vytváření textových analyzátorů. Modul PostUtils obsahuje funkce a třídy pro vytváření postprocesorů. PathUtils modul slouží pro práci s interním kódem Path modulu (CL daty). Na závěr je importován modul datetime a definována proměnná now, která obsahuje aktuální čas.

```
#Import modulů
#-------------------------------------------------
from __future__ import print_function
import FreeCAD
import Path
from FreeCAD import Units
import shlex
from PathScripts import PostUtils
from PathScripts import PathUtils
import datetime
now = datetime.datetime.now()
#-------------------------------------------------
```

Obr. 40: Import modulů

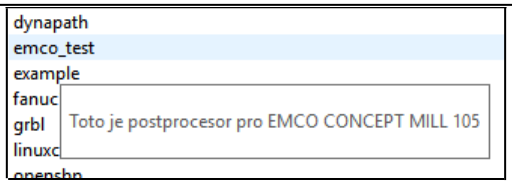
Po importu modulů jsou následující částí kódu „Uživatelsky nastavitelné hodnoty“. Kde si uživatel může nastavit některé parametry a ovlivnit tak finální výstup postprocesoru. Patří sem například zobrazování informativní hlavičky programu, jejíž obsah je popsán dále. Při zadání hodnoty pravda (*True*) je hlavička zobrazena, při hodnotě nepravda (*False*) není vypsána. Dále zda má být výstupní kód modální, což znamená že nezobrazuje stejné hodnoty jako jsou na předchozím řádku a zlepšuje tak orientaci v ISO kódu. Následující parametr *CISLA_RADKU* při hodnotě True zajišťuje přidání čísla řádku na každý řádek programu. Proměnná *Radek* určuje od které hodnoty bude začínat číslování jednotlivých řádků. Další segment obsahuje informace které jsou vypsány před převodem CL dat. Tímto segmentem je *Uvodprg*. Ten obsahuje informace, které nejsou obsaženy v CL datech a je potřeba je pro správné fungování programu doplnit. Trojúvozovky naznačují, že jde o víceřádkový textový řetězec a v cílovém souboru sou tak zobrazeny jen zeleně označené informace. *Konecprg* funguje na stejném principu akorát s tím rozdílem že je vypsán na samotném konci programu. *Komentar* nahrazuje původní formát komentáře v kulatých závorkách za hodnotu zadanou jako řetězec mezi uvozovkami, což je v tomto případě středník.

```
#Uživatelsky nastavitelné hodnoty
#-------------------------------------------------
Hlavicka = True #True-Vypíše úvodní část programu, False - Nevypisuje
MODAL = True    #Pokud je hodnota True neopakuje příkazy se stejnou
CISLA_RADKU = True  # True - vypisuje čísla řádku Nxxx / False - nevypisuje
Radek = 0 # Počatečni hodnota pro čislovani radku
#Úvodní část programu
Uvodprg = '''; Uvodni cast programu
G17 G40
G54 G90
'''
#Konečná část programu
Konecprg = ''' M30
'''
Komentar = ";" # Komentář pro Sinumerik
#-------------------------------------------------
```

Obr. 41: Nastavitelné uživatelské hodnoty

Následující úsek kódu na horní části obrázku (Obr. 42) slouží, k zobrazení nápovědy (*Tooltip*) při výběru postprocesoru. Zobrazuje se pouze text mezi trojúvozovkami. K tomu aby se vůbec postprocesor zobrazil v nabídce funkce FreeCADu „Postprocess“ musí být jeho zdrojový kód uložen do složky s ostatními postprocesory, jak již bylo zmíněno viz. kapitola (3.2). Poté se při najetí kurzoru na název zobrazí zeleně označený text.

```
#TOOLTIP-Napoveda při výběru postprocesoru v funkci JOB
TOOLTIP='''
Toto je postprocesor pro EMCO CONCEPT MILL 105
        '''
```



Obr. 42: Nápověda

Přiřazení proměnné *pythonopen* k vestavěné funkci pythonu pro otevření souboru (o*pen*) je vidět na obrázku (Obr. 43). Podmínka *if* je v tomto případě vždy pravda, protože souborem použitím pro postprocesing je momentálně otevřený soubor v Path modulu. Tento zjednodušený zápis je pak využit dále při otevření souboru pro zápis finálního výstupu.

```
# otevření souboru
if open.__module__ == '__builtin__':
    pythonopen = open
```

Obr. 43: Funkce otevření souboru

První definovanou funkcí je *cislaradku*, jejíž definici je možné vidět na obrázku (Obr. 44). Ta obsahuje vytvoření proměnné *Radek*. Poté následuje podmínka zda je hodnota *CISLA_RADKU* pravda. Ta se nastavuje v části uživatelský proměnných viz (Obr. 41). Poté je proměnná *Radek* zvětšena o hodnotu 10. Operator „+=“ pracuje stejně jako kdyby bylo napsáno *Radek = Radek +10*. Poté funkce vrací (*return*) textový řetězec složený ze tří částí. První z nich je písmeno N napsané ve tvaru „N“. To značí že jde o obyčejný textový řetězec. Následující částí je samotná hodnota proměnné která, je ovšem celé číslo [15] a je tak nutné ho pomocí příkazu str()[16] převést na textový řetězec. V pythonu totiž nelze spojovat řetězcové a číselné hodnoty jinak python vypisuje následující chybu: „*cannot concatenate 'str' and 'int' objects*“ do zobrazení reportu FreeCADu. Poslední částí řetězce je mezera která, je brána jako řetězec mezi dvěma uvozovkami. Všechny tyto části jsou spojeny (*concatenate*) operátory „+“, které z nich dělají jeden řetězec. V případě že podmínka *if* neplatí je výstup prázdný řetězec „“, bez mezery.

```
#čísla řádku
def cislaradku():
    global Radek
    if CISLA_RADKU == True:
        Radek += 10 # hodnota o kterou se zvetsi následující cislo radku
        return "N" + str(Radek) + " "
    return ""
```

Obr. 44: Funkce pro čísla řádků

Na následujícím obrázku (Obr. 45) se nachází první ze čtyř částí exportní funkce (*export*), která je hlavní částí programu. Vstupními parametry této funkce jsou *objectlist*, což je seznam všech objektů FreeCADu, dále *filename,* což je zadaný název souboru ve funkci Job. Posledním parametrem je řetězec pro případné argumenty (*argstring*[17]), které mohou upravovat parametry postprocesoru a ovlivnit tak konečný výstup. Ovšem pouze v případě že postprocesor těmito řetězcovými argumenty disponuje. Navržený postprocesor argumenty nedisponuje ale funkce export je již zadefinována ve FreeCADu s těmito třemi vstupními argumenty a toto je v podstatě úprava její definice. Proto musí být zadány všechny tři vstupní parametry i když případě argstring jde vždy o prázdnou hodnotu, jinak FreeCAD vypíše chybové hlášení: „*export() takes exactly 2 arguments (3 given)*“. Následující smyčka *for-in* zde zajišťuje, že všechny vstupní objekty (obj) z *objectlist* mají atribut (*hasattr – has Attribute*) Path a ostatní objekty jsou vyřazeny neboť pro finální výstup nejsou potřeba. Následně je do zobrazení reportu FreeCADu vypsáno „Probíhá postprocessing…“ a je vytvořena proměnná *vystup*. Ta je v tento moment prázdná ale právě do ní budou postupně přidávány jednotlivé informace pro finální výstupní soubor.

[15] Integer – Celé číslo značeno v Pythonu jako int
[16] String - řetězec
[17] Argument string

```
def export(objectslist, filename,argstring):
    for obj in objectslist:
        if not hasattr(obj, "Path"):
            return None
    print("Probíha postprocessing...")
    vystup = ""
```

Obr. 45 Funkce export – Definice (část 1/4)

Obrázek (Obr. 46) zobrazuje druhou část exportní funkce. Zde jsou do proměnné *vystup* přidávány jednotlivé informace. Pokud je v uživatelských proměnných definována *Hlavicka* jako pravda jsou do výstupu přidány informace se jménem postprocesoru a na dalším řádku čas exportu ve formátu: „Den. Měsíc. Rok Hodina: Minuta: Vteřina“. Následně je z víceřádkového řetězce *Uvodprg,* jehož obsah je zadefinován v uživatelských proměnných, vypsán každý řádek (*line*) a k němu přidané číslo řádku. Tomu je tak díky metodě pro práci s řetězci *splitlines,* která vrací veškeré řetězce jako seznam řádků. Zápis v *Uvodprg* by mohl být také zapsán na jednom řádku a vypadat pak následovné: „; Uvodni cast programu\n G40 G17\nG54 G90“. Výsledek by byl v tomto případě stejný neboť značka „\n“ signalizuje nový řádek. Pokud by ovšem *splitlines* byla nepravda byly by všechny řetězce vypsány jako jeden řádek. Následující cyklus *for-in* je opět pro objekty obsažené v *objectlist* ale v tomto případě jsou to pouze objekty které mají atribut Path. V tomto cyklu jsou dále do proměnné výstup vkládána data z další funkce, kterou je *parse*, jejíž význam je popsán následovně. Poslední částí přidané do proměnné výstup je část *Konecprg*, která funguje na stejném principu jako výše popsaný *Uvodprg*.

```
if Hlavicka:
    vystup += cislaradku() + ";Export souboru postprocesorem: " + __name__ + " \n"
    vystup += cislaradku() + ";Cas exportu: " + str(now.strftime('%d.%m.%Y %H:%M:%S')) + "\n"
for line in Uvodprg.splitlines(True):
    vystup += cislaradku() + line
for obj in objectslist:
    vystup += parse(obj)
for line in Konecprg.splitlines(True):
    vystup += cislaradku() + line
```

Obr. 46: Funkce export - Vystup (část 2/4)

Předposlední část funkce export je na obrázku (Obr. 47). Jde o zobrazení interního editoru G-kódu FreeCADu. Ten bylo možné vidět například na obrázku (Obr. 15) v kapitole 3.4.3. Výstup je tak zobrazen v tomto editoru a pokud uživatel provede změny které potvrdí OK je výstup uložen v upravené formě z editoru do proměnné *final*. V opačném případě je uložen původní kód který byl zobrazen v editoru z proměnné *vystup*.

```
if FreeCAD.GuiUp:
    dia = PostUtils.GCodeEditorDialog()
    dia.editor.setText(vystup)
    result = dia.exec_()
    if result:
        final = dia.editor.toPlainText()
    else:
        final = vystup
```

Obr. 47: Funkce export - Editor G-kódu (část 3/4)

Poslední část exportní funkce je zobrazena na obrázku (Obr. 48). Zde je první příkaz vypsání textového řetězce „Posprocessing hotov.“ do zobrazení reportu FreeCADu. Dále je vytvořen soubor (objekt) *gfile* pomocí dříve zadefinované funkce pythonopen[18], která má dva parametry. Prvním parameterm je název souboru (*filename*), druhý parametr „w“ značí způsob jakým python bude soubor používat. V tomto případě je to hodnota pro zápis (*write*). Standardní hodnotou je „r“(*read*), což značí pouze čtení souboru v případě že soubor již existuje. Následně jsou do souboru zapsány (*write*) informace z proměnné *final* a poté je soubor zavřen (*close*).

```
print("Postprocessing hotov.")
#Otevření okno pro uložení
gfile = pythonopen(filename,"w")
gfile.write(final)
gfile.close()
```

Obr. 48: Export funkce - Uloženi souboru (část 4/4)

Na následujícím obrázku (Obr. 49) je první část již zmíněné funkce *parse*, jejíž vstupními parametry jsou objekty Path modulu (*pathobj*). Název funkce pochází z anglického slova *Parsing* což lze přeložit jako *Parsování.* To je v informatice chápáno jako syntaktická analýza. Tedy analýza textu, která zde v tomto případě slouží k překladu vstupních informací v podobě objektů Path modulu na výstupní ISO kód. Nejprve je zadefinována globální proměnná *MODAL*, která má v uživatelském nastavení přiřazenou hodnotu *True.* Dále jsou zadefinované proměnné *out*, což je výstupní proměnná do které jsou postupně ukládány přeložené řádky. Další vytvořenou proměnnou je *posledni_prikaz*, sloužící k uložení posledního příkazu který prošel funkcí *parse*, což je pak dále využito pro zajištění modality[19] výstupního kódu. Následně jsou zadefinovány dva seznamy. První je seznam všech parametrů, které se mohou v CL datech vyskytnout. Další seznam obsahuje parametry které mohou být ve výsledném kódu modální. Pak je založena proměnná *modal_param* kde vlnité závorky naznačují že jde o datový typ slovníku. Ten se skládá ze seznamu dvojic hodnot, které jsou ve formátu klíč-hodnota. Klíč je datového typu řetězec a hodnota číslem, tedy například „X“: 10. Předposledním úkonem je definice objektu *prvnihodnota*, který slouží pro vytvoření fiktivní hodnoty pro porovnávání se samotnými hodnotami CL dat. Poté jsou do slovníku *modalparam* přidány parametry z objektu *prvnihodnota*. Zajímavou vlastností slovníku v pythonu je že při přiřazení klíče je automaticky přiřazena i jeho hodnota. To znamená že z objektu *prvnihodnota* jsou přiřazeny parametry X, Y atd. a to včetně jejich přidělených hodnot.[20]

```
def parse(pathobj):
    global MODAL
    out = ""
    posledni_prikaz = None
    parametry = ['X', 'Y', 'Z', 'A', 'B', 'I', 'J', 'F', 'S', 'Q', 'R', 'L', 'H','T']
    modal_parametry = ['X', 'Y', 'Z','A', 'B', 'F', 'S', 'T']
    modalparam = {}
    prvnihodnota = Path.Command ("G0", {"X":-1, "Y":-1, "Z":-1, "F":0.0, "T":0})
    modalparam.update(prvnihodnota.Parameters)
```

Obr. 49: Funkce Parse - definováni (část 1/4)

Další část funkce *parse* je na obrázku (Obr. 50). Zde je další podmínka, která má sloužit k filtrování vstupních objektů Path (*pathobj*) které mají atribut skupiny (*Group).* Ta slouží

[18] Pythonopen – zadefinováno z funkce open(jméno_souboru, parametry)
[19] Stejné hodnoty nejsou na dalším řádku zobrazeny.

k tomu pokud by si uživatel vytvořil podsložku pro jednotlivé operace ve stromovém zobrazení. Podmínka *for-in* slouží k tomu aby každý objekt v této skupině prošel funkcí *parse*. Následně pokud objekt nemá atribut skupiny a jedná se o samostatnou dráhu, program tento úsek přeskakuje a jde rovnou do části *else* této funkce. Kde je pro každý osamocený objekt (řádek) vytvořen výstupní seznam *outstring*. Příkazům jejichž hodnota je G, M či komentář je přiřazená proměnná *command.* Poté pokud je hodnota command nulová a jedná se tedy o komentář, tak ten je z původního tvaru, kterým je text v závorkách nahrazen středníkem. Následně je tento příkaz přidán do seznamu *outstring* jako poslední hodnota (a*ppend* – připojit) v případě příkazu je to první pozice. Pokud je hodnota *MODAL* zadefinována jako pravda a jsou příkazy, které jsou uloženy jako poslední příkaz totožné s právě načteným příkazem *command,* dojde k smazání první pozice v seznamu *outstring*, což je právě *command*.

```
if hasattr(pathobj, "Group"):  # parsovani skupiny drah
    for p in pathobj.Group:
        out += parse(p)
    return out
else:
    for c in pathobj.Path.Commands:
        outstring = []
        command = c.Name #Jmena jednotlivých příkazu z Command obj G,M,Komentar
        if command[0]=='(': #Koemntař ;
            command = PostUtils.fcoms(command, Komentar)
        outstring.append(command) # Přidaní příkazu
        if MODAL is True: #Neopakuje stejne priakzy (G,M)
            if command == posledni_prikaz:
                outstring.pop(0)
```

Obr. 50: Funkce Parse - Příkazy (část 2/4)

Pokračování funkce *parse* je na obrázku (Obr. 51). Po uložení příkazu *command* jsou na řadě parametry bloku. Pořadí parametru je definováno jejich pozicemi v seznamu *parametry* (Obr. 49). Cyklus *for param in parametry* probíhá pro každý parametr momentálně načteného objektu. Poté následuje série podmínek pro jednotlivé parametry. Prvním podmínka se týká parametru posuvu F. Pokud je parametr F a není stejný jako parametr F uložení v *modalparam* je vypsán do řetězce *outstring* ale pouze v případě, že jeho hodnota je větší než 0. FreeCAD totiž vypisuje hodnoty jak pro pracovní tak i pro rychloposuv ovšem ten má v tomto případě zadanou hodnotu 0 a není tak vypisován. Tomuto navíc ještě předchází podmínka, která se ptá zda příkaz není G0. Dále bylo nutné provést převod hodnoty posuvu, protože FreeCAD standardně vypisuje posuv v milimetrech za vteřinu (mm/s), což ji přinejmenším nestandardní. Proto je nutné jednotky převést na milimetry za minutu (mm/min). Hodnota je pak vypsána do listu *outstring* ve formátu param (F) + jeho zformátovaná číselná hodnota pomocí metody format(). Tato metoda má dva parametry: řetězec, který má být formátován a argumenty formátování. V případě parametru F bylo nutné, přesto že výsledkem má být celé číslo, nastavit formát hodnoty na *float* (desetinné číslo) s nulovým počtem čísel za desetinou tečkou pomocí parametru.*0f.* Bylo tak nutné učinit z toho důvodu, protože jinak FreeCAD zaokrouhluje převedené hodnoty a namísto posuvu F200 bylo vypsáno F199 při nastavení hodnoty int. V případě že se jedná o parametr F ale není splněna podmínka modality není tak posuv vypsán aby nebyl výstup zbytečně nepřehledný a cyklus pokračuje na další parametr. Tím je parametr středu pro kruhovou interpolaci I, který je vypsán do seznamu *outstring* obdobně jako parametr F ale zde je jeho hodnota zformátována na tři desetinná čísla za tečkou a to pomocí argumentu.*3f* metody format(). Naprosto stejně je to i v případě parametru J. Následující parametry T a S jsou řešeny obdobně ale s tím rozdílem že jsou

vypsány jako celé číslo. Další podmínka zajišťuje, že parametr Y bude vypsán v případě, že se jedná o kruhovou interpolaci a parametr Z není totožný jako poslední uložený parametr Z. To znamená že se jedná o prostorovou kruhovou interpolaci. Poseldní podmínka spadající do cyklu kontroluje zda je aktuální parametr totožný s již nahraným parametrem v *modalparam.* Pokud tomu tak není je parametr vypsán ve formátu s třemi desetinnými místy.

```
for param in parametry:
    if param in c.Parameters:
        if param == 'F'and (modalparam[param] != c.Parameters[param]):
            if c.Name not in ["G0", "G00"]:
                speed = Units.Quantity(c.Parameters['F'], FreeCAD.Units.Velocity)
                if speed.getValueAs('mm/min') > 0.0:
                        outstring.append(param + format(float(speed.getValueAs('mm/min')),'.0f' ))
                else:
                    continue
        elif param == 'I':
            outstring.append(param + format(c.Parameters['I'], '.3f'))
        elif param == 'J':
            outstring.append(param + format(c.Parameters['J'], '.3f'))
        elif param == 'T':
                outstring.append(param + format(int(c.Parameters["T"])))
        elif param == 'S':
                outstring.append(param + format(int(c.Parameters[param])))
        elif param =="Y" and (c.Name in ["G2", "G3"])and (modalparam["Z"] != c.Parameters["Z"]):
                outstring.append(param + format(c.Parameters[param], '.3f'))
        else:
            if (param in modalparam) and (modalparam[param] == c.Parameters[param]):
                continue
            else:
                outstring.append(param + format(c.Parameters[param], '.3f'))
```

Obr. 51: Funkce Parse - Parametry (část 3/4)

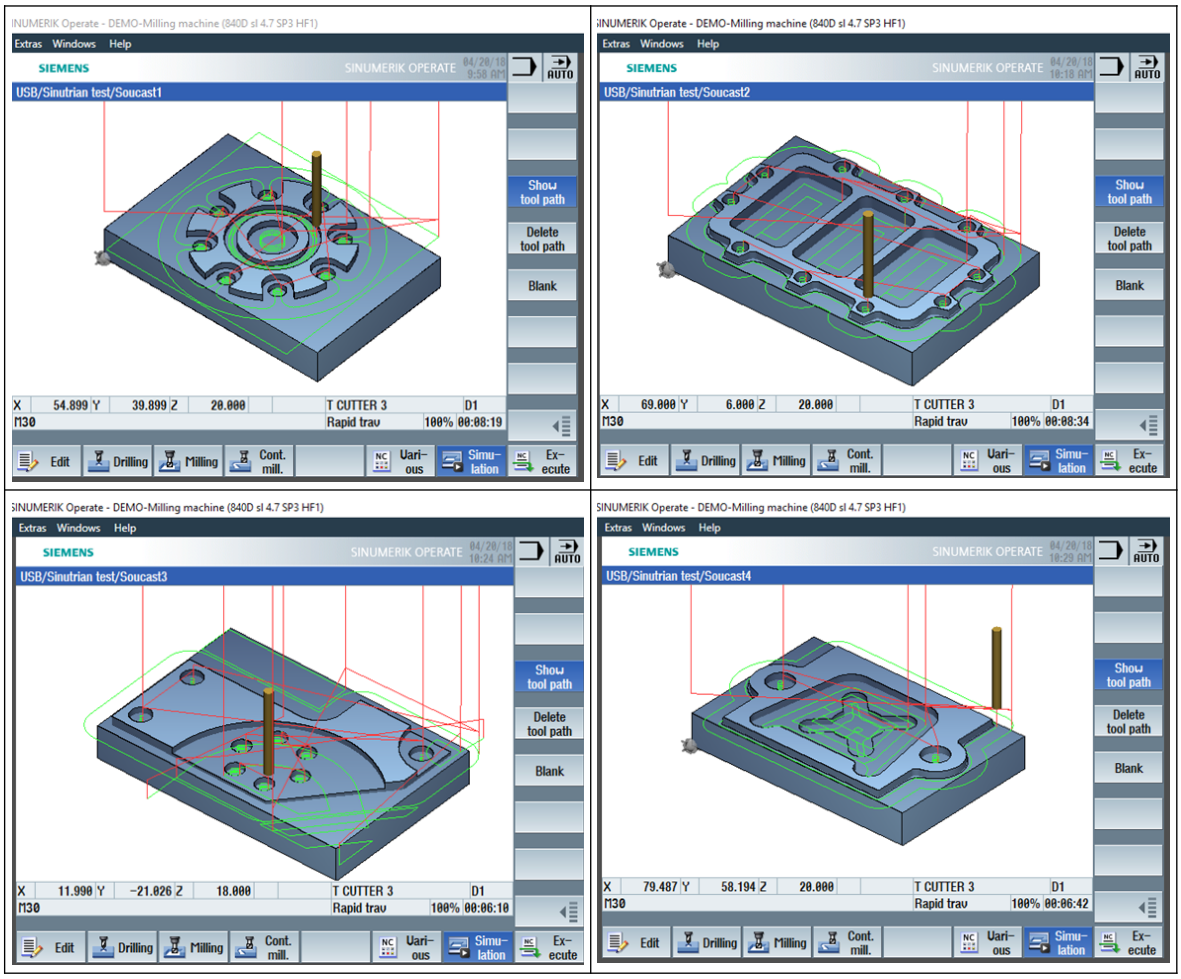
Poslední část funkce *parse* se nachází na obrázku (Obr. 52). Do proměnné *poslední_prikaz* je uložen při každém průchodu funkce nový příkaz command a slovník *modalparam* je aktualizován o nové hodnoty parametrů. Následuje podmínka pokud je příkaz roven hodnotě M6, tak je přidán do výstupu komentovaný řetězec „;Vymena nastroje“. Poslední podmínkou ve funkci *parse* je v případě že je velikost výstupního listu *outstring* větší než jedna, tedy obsahuje li alespoň jednu hodnotu je na první místo přidáno číslo řádku pokud ovšem pouze pokud je *CISLA_RADKU* pravda. Následně je každá hodnota ve výstupním listu *outstring* vložena do proměnné *out,* která je výstupem z funkce *parse* a ta tuto hodnotu vrací funkci export.

```
            # ulozeni posledniho prikazu a modalniho parameteru
            posledni_prikaz = command
            modalparam.update(c.Parameters)

            # Prida retezec vymnena nastroje
            if command == 'M6':
                    out += cislaradku() + ";Vymena nastroje\n"

            # Cislo bloku jako prvni hodnota
            if len(outstring) >= 1:
                if CISLA_RADKU:
                    outstring.insert(0, (cislaradku()))

                # Pridani radku do finalniho vystupu
                for w in outstring:
                    out += w + " "
                out = out.strip() + "\n"

        return out
```

Obr. 52: Funkce Parse - Výstup (část 4/4)

5. Ověření funkce postprocesoru

Funkčnost jednotlivých výstupů z FreeCADu převedených nově vytvořeným emco postprocesorem (emco_post.py) proběhla nejprve pomocí software SinuTrain. Ten simuluje prostředí řídicího systému a je v něm možné simulovat vytvořené programy stejně, jako kdyby uživatel byl přímo u stroje. Úspěšně dokončené simulace programů pro jednotlivé vzorové součásti je možné vidět na obrázku (Obr. 53). Výsledek v tomto simulovaném prostředí byl takový jaký se očekával. To sice že dráhy jsou totožné se simulací ve FreeCADu. Z toho je možné usoudit že programy by měly s největší pravděpodobností fungovat stejně i na stroji. Při zkoušce programů v SinuTrainu bylo zjištěno že komentáře v ISO kódu musí být značeny středníkem jako prvním znakem po čísle řádku.



Obr. 53: Test programů SinuTrain

Poté byl kód ověřen přímo na stroji, který je možné vidět na obrázku (Obr. 54) po rekonstrukci a změně řídicího systému. Při porovnání s předešlým stavem na obrázku (Obr. 17 v kapitole 3.6) je na první pohled patrná změna především v absenci počítače a přidání ovládacího panelu s řídicím systémem Sinumerik 828D.

Obr. 54: Stroj po rekonstrukci

Bohužel v době dokončování práce, nebyl stroj ještě zcela připraven pro otestování programů výrobou součástí. Proto byly programy z organizačních a technických důvodů vyzkoušeny pouze simulací přímo na stroji, která zajišťuje že stroj bude schopen kód přečíst a součást vyrobit. Teprve zde bylo zjištěno několik nedostatků kódu který postprocesor vyprodukoval a které se v simulaci SinuTrainu neobjevily. První nedostatkem byla absence parametru Y při pohybu nástroje po šroubovici, což se vyskytuje v programech s použitím obráběcí strategie helix. Důvodem bylo že parametr Y je brán v kódu jako modální parametr. Ovšem funkce kruhové interpolace při pohybu v prostoru potřebuje v bloku veškeré parametry. Naproti tomu pokud je kruhová interpolace v rovině a parametr Z je modální blok proběhne bez problému. K napravení tohoto nedostatku byla ve funkci *parse* přidána podmínka: „*elif param =="Y" and (c. Name in ["G2", "G3"])and (modalparam["Z"]!= c. Parameters["Z"])*“ (viz.Obr. 51) která zajišťuje vypisování stejných příkazů Y v případě že se nachází na řádce s příkazem G2 či G3 a za předpokladu že předchozí hodnota parametru Z je odlišná od následující hodnoty. V následující tabulce (Tab. 12) je ukázka dvou operací s kruhovou interpolací s tím že v levé části je kruhová interpolace v rovině a parametr Z je modální a tudíž není v bloku vypsán. V pravé části je kruhová interpolace v prostoru, kdy se nástroj pohybuje po spirále a je nutné tak vypisovat parametr Y i přesto že je vždy stejný jinak Sinumerik zahlásí chybu se špatnou definicí středu kružnice.

Kruhová interpolace v rovině	Kruhová interpolace v prostoru
N230 G2 X46.211 Y66.477 I26.149 J25.461 N240 X81.477 Y28.789 I-1.211 J-36.477 N250 X43.789 Y-6.477 I-36.477 J1.211 N260 X18.851 Y4.539 I1.211 J36.477	N1060 G2 X41.000 Y30.000 Z14.583 I-4.000 J0.000 N1070 X49.000 Y30.000 Z14.167 I4.000 J0.000 N1080 X41.000 Y30.000 Z13.750 I-4.000 J0.000 N1090 X49.000 Y30.000 Z13.333 I4.000 J0.000

Tab. 12: Výstupy pro kruhové interpolace v rovině a v prostoru

Dalším nedostatkem bylo že funkce M30 nebyla následována prázdným řádkem a řídicí systém tak hlásil chybu nesprávného ukončení programu. To bylo ošetřeno jednoduchým odřádkováním ve víceřádkové řetězcové proměnné *Konecprg*. Alternativně by zápis mohl vypadat také jako:“ ““M30\n““““ se stejným výsledkem.

Po ošetření těchto nedostatků proběhlo ozkoušení programu vytvořeného pro součást číslo 1, již bez chybových hlášení. Výsledek je možné vidět na obrázku (Obr. 55). Simulace proběhla stejně úspěšně jako v simulaci drah ve FreeCADu tak v software SinuTrain. Z toho lze soudit že výstup z postprocesoru je správný a jednotlivé součásti by bylo možné

vyrobit, což nakonec nebylo realizováno. Další programy nebylo nutné již simulovat protože obsahují operace stejného charakteru a je tak jisté že budou fungovat stejně.

Obr. 55: Ověření programu na stroji

6. Zhodnocení a závěr

Postprocesory jsou vytvářeny jakožto nezbytná součást či externí doplněk CAM systému použitého pro výrobu. Bez postprocesoru je jakýkoliv CAM systém pouhým simulátorem pohybu nástroje po drahách vytvořených pomocí funkcí v daném systému. Výše popsaný, vytvořený postprocesor „emco“ lze považovat jako příspěvek k vývoji software FreeCAD, konkrétně k jeho části zabývající se vytvářením drah nástroje, tedy Path modulu.

Samotné tvorbě postprocesoru předcházelo několik kroků. Prvním krokem před samotným řešením problému bylo specifikování problému, definování cílů a popis okolností vstupujících do problematiky tvorby postprocesoru. Druhým krokem byla rozsáhlá analytická část zabývající se jak obecnou problematikou postprocesorů, tak také situací týkající se konkrétně FreeCADu a jeho postprocesorů. Poté byly popsány důležité části, klíčového činitele práce, kterým je FreeCAD. Těmito částmi jsou PartDesign (CAD) a Path (CAM) modul. Mnohem detailněji byl probádán modul Path, kde byly prozkoumány kromě jednotlivých funkcí obráběcích strategii i způsoby zobrazení CL dat FreeCADu, což je klíčovou informací pro postprocesor. Součástí analytické části je také samotný programovací jazyk Python. Kde nešlo o rozsáhlý popis tohoto jazyka, ale pouze o přiblížení jeho podstaty. Posledním bodem analýzy byl samotný stroj EMCO CONCEPT MILL 105 a ISO kód.

Po analytické části nasledoval již popis samotné tvorby postprocesoru s vysvětlením jednotlivých částí programu. Výstupy z postprocesoru byly pak otestovány ve dvou variantách. První varianta testování proběhla pomocí systému SinuTrain, neboť stroj byl v daném časovém období na rekonstrukci a změně řídicího systému. Následně po příchodu stroje byla funkčnost programů otestována přímo simulací na stroji. Bohužel stroj nebyl ještě připraven na otestování programů výrobou součástí, a proto byla provedena pouze simulace programů. V obou případech byly objeveny nedostatky ve výstupním kódu, ovšem ty se podařilo odstranit úpravou kódu postprocesoru a poté byla validace jednotlivých programů úspěšná.

Při tvorbě programů v Path modulu pro jednotlivé vzorové součásti byl souběžně vytvořen ještě program pomocí ručního vytváření kódu pro jednu ze zvolených součástí. Důvodem pro tento úkon bylo zjištění orientační časové úspory při použití bezplatného CAM systému oproti ručnímu vytváření programu. V obou případech bylo jako podklad pro testování poskytnut pouze výkres součásti. Při vytváření kódu ručně byl program vytvořen za 86,3min (1: 26: 18). Při tvorbě programu pomocí FreeCADu byl celkový čas 23,8 minuty (0: 23: 48), kde cca 9 minut zabrala tvorba modelu a zbytek času připadl na tvorbu technologie. Poté by v obou případech připadl ještě čas na odladění programu pro korektní výrobu součásti. Při porovnání se FreeCAD jeví jako lepší varianta. Časový rozdíl by s rostoucí komplexností jednotlivých součástí rostl, a výsledek by byl také ve prospěch FreeCADu. Další využití FreeCADu by mohlo být například při součástech s parametrickými prvky, kde by stačilo vzít soubor s již vytvořeným programem a upravit model. Dráhy nastroje se poté následně upraví dle hodnot modelu, což by byl rychlejší proces než nová tvorba kódu ruční variantou.

Vytvořený postprocesor lze tak považovat za funkční. Podnět k jeho tvorbě byl především z toho důvodu, že všechny momentálně dostupné postprocesory FreeCADu produkují výstup, který je nevhodný pro řídicí systémy Sinumerik, a není tedy možné využít žádný z nich pro vytváření programů se zaměřením na stroje s řídicím systémem Sinumerik. Vytvořený postprocesor lze využít i na ostatní stroje podobného typu (tříosá frézka) s řídicími systémy Sinumerik, avšak zatím pouze pro operace 2,5D a částečně 3D frézování.

Samotný kód postprocesoru nelze považovat za perfektní a dokonce ani ve finálním stavu jeho vývoje. Stále je zde místo pro další vylepšování, aby tak bylo možné využívat plný potenciál stroje. Jeho největším nedostatkem jsou vrtací cykly. Bohužel jejich převod nebyl nakonec do kódu postprocesoru zakomponován a postprocesor tak stále generuje výchozí nevhodné cykly G81-G83 namísto cyklů sinumeriku pro vrtání (CYCLE81-83). FreeCAD se současným postprocesorem tedy zatím není možné využít pro vrtání na cílovém stroji. Pokud by došlo k vydání další funkce FreeCADu, která by operaci vrtání produkovala jako lineární interpolaci pohybů (G0, G1) bylo by možné použít stávající postprocesor. Tento problém tak zůstává námětem pro další vývoj postprocesoru a případně i Path modulu. Ten má před sebou ještě dlouhou cestu pro naplnění finální vize vývojářů a poskytnutí všech služeb v plném rozsahu koncovým uživatelům, jak by se od CAM systému očekávalo.

Nehledě na to, že FreeCAD je stále ve fázi vývoje a jeho část pro obrábění (Path modul) je ve velmi raném stadiu vývoje, se FreeCAD celkově jeví jako zajímavý CAx systém, a ze získaných informací lze bez zaváhání říci, že je zde odůvodněná naděje k tomu, aby se tento software stal v budoucnu kvalitním a rozšířeným CAx systémem.

Seznam použité literatury

[1] **VINTR, Lukáš**. *Tvorba vzorových příkladů pro výuku NC programování pro nerotační součásti*. Plzeň, 2010. Bakalářská práce. Západočeská Univerzita V Plzni. Fakulta strojní. Katedra technologie obrábění Vedoucí práce Jiří ČESÁNEK

[2] **FreeCAD**. *FreeCAD wiki - Main Page*. [Online]. 8.10.2017 [Cit. 20.11.2017] Dostupné z: https://www.freecadweb.org/wiki/Feature_list

[3] **Python Software Foundation**. *The Python Logo*. [Online]. 2017 [Cit. 20.11.2017] Dostupné z: https://www.python.org/community/logos/

[4] **Sadílek, Miroslav**. *Postprocesor - slabé místo CAM systému ?* MM Spektrum. [Online]. 13.04.2005 [Cit.20.11.2017]
Dostupné z: https://www.mmspektrum.com/clanek/postprocesor-slabe-misto-cam-systemu.html

[5] **Free Software Foundation, Inc.**. *What is free software?* [Online]. 01.01.2018 [Cit. 25.01.2018] Dostupné z: https://www.gnu.org/philosophy/free-sw.html.en

[6] **Tovarna**. *Pojem "LGPL"*. [Online]. 2018 [Cit. 25.01.2018]
Dostupné z: http://www.tovarna.cz/cz/slovnik-pojmu/44-lgpl/

[7] **Keller, Petr**. *Postprocesory a ukázka tvorby postprocesoru*. [Online]. 9.12.2011 [Cit. 20.11.2017]
Dostupné z: http://educom.tul.cz/educom/inovace/PNC/VY_03_16-postprocesory%20a%20uk%C3%A1zka%20tvorby%20postprocesoru_p%C5%99_MZ_6.pdf

[8] **LERCH, Jan**. *Tvorba postprocesoru pro sinumerik 840 ve vybraném CAD/CAM systému*. Plzeň, 2010. Bakalářská práce. Západočeská Univerzita v Plzni. Fakulta strojní. Katedra technologie obrábění. Vedoucí práce Karel JANDEČKA

[9] **KOŽMÍN, Pavel**. *Metoda tvorby víceosého postprocesoru*. Plzeň, 2004. Disertační práce. Západočeská univerzita v Plzni. Fakulta strojní. Katedra technologie obrábění. Vedoucí práce Karel Jandečka

[10] **Robodoupě**. *CNC jednoduše – 1. část: Co je Grbl?* 13.8.2013. [Online]. [Cit. 28.1.2018] Dostupné z: http://robodoupe.cz/2013/cnc-jednoduse-1-cast-co-je-grbl/

[11] **FreeCAD**. *Path scripting*. [Online]. 06.03.2018 [Cit. 10.03.2018] Dostupné z: https://www.freecadweb.org/wiki/Path_scripting#FreeCAD.27s_internal_GCode_format

[12] **FreeCAD**. *Mouse Mode*. [Online]. 8.10.2017 [Cit. 25.01.2018] Dostupné z: https://www.freecadweb.org/wiki/Mouse_Model

[13] **FreeCAD**. *Path Workbench*. [Online]. 8.10.2017 [27.01.2018] Dostupné z: https://www.freecadweb.org/wiki/Path_Workbench

[14] **Skoupil, David**. *ÚVOD DO PARADIGMAT PROGRAMOVÁNÍ* . [Online]. 2007 [Cit. 23.3.2018] Dostupné z: https://phoenix.inf.upol.cz/esf/ucebni/uvod_para.pdf

[15] **Tutorialspoint**. *Python-Functions*. [Online]. 2018 [Cit. 15.3.2018] Dostupné z: https://www.tutorialspoint.com/python/python_functions.htm

[16] **EMCO GmbH**. *CONCEPT MILL 105*. [Online]. 2016 [Cit. 8.10.2017] Dostupné z: https://www.emco-world.com/en/products/industrial-training/machines/milling/cat/26/d/2/p/10000452C26/pr/concept-mill-105.html

[17] **KTO, ZČU**. Výuková *laboratoř NC programování - UL211*. [Online]. 2018 [Cit. 20.1.2018] Dostupné z: https://www.kto.zcu.cz/o-katedre/Vybaveni_katedry/UL211.html

[18] **SOŠ Jana Tiraye**. *Obecný úvod do problematiky CNC programování*. [Online]. 2017 10.12.2017
Dostupné z http://www.sosbites.cz/images/stories/Pro-studenty/studijni-materialy/VUKOV_TEXT_-_1.ST.pdf

[19] **Technor**. *ČSN ISO 6983-1 (184315)*. [Online]. 2018 [Cit. 20.4.2018] Dostupné z: http://www.technicke-normy-csn.cz/184315-csn-iso-6983-1_4_25310.html

[20] **The Python Standard Library**. *Built-in Types*. [Online]. 2018 [Cit. 27.4.2018] Dostupné z: https://docs.python.org/3/library/stdtypes.html#typesmapping

Seznam ilustrací

Seznam příloh

Přílohy všité do vazby

Příloha č.1 – Kód postprocesoru
Příloha č.2 – CL data – ukázka (Součást č.1)
Příloha č.3 – Převedený ISO kód – ukázka (Součást č.1)

Přílohy volně vložené (Výkresová dokumentace) [1]

Výkres vzorové součásti č.1 (č.v. BP KTO 10-1)
Výkres vzorové součásti č.2 (č.v. BP KTO 10-2)
Výkres vzorové součásti č.3 (č.v. BP KTO 10-18)
Výkres vzorové součásti č.4 (č.v. BP KTO 10-19)

PŘÍLOHA č. 1

Kompletní kód postprocesoru

emco_post.py - C:\Users\Martin\Desktop\FreeCAD_0.17.12570_x64_dev_win\Mod\Path\PathScripts\post\emco_post.py (3.6.4)

File Edit Format Run Options Window Help

```
#encoding: utf -8
#Postprocesor pro EMCO CONCEPT MILL 105
#Vytvoren v rámci DP: "Postprocesor pro zvoleny stroj k SW FreeCAD"
#vychazi z linuxCNC_post.py

#Import modulů
#----------------------------------------------------
from __future__ import print_function
import FreeCAD
import Path
from FreeCAD import Units
import shlex
from PathScripts import PostUtils
from PathScripts import PathUtils
import datetime
now = datetime.datetime.now()
#----------------------------------------------------

#Uživatelsky nastavitelné hodnoty
#----------------------------------------------------
Hlavicka = True #True-Vypíše úvodní část programu, False - Nevypisuje
MODAL = True    #Pokud je hodnota True neopakuje příkazy se stejnou
CISLA_RADKU = True  # True - vypisuje čísla řádku Nxxx / False - nevypisuje
Radek = 10 # Počatečni hodnota pro číslovani radku
#Úvodní část programu
Uvodprg = '''; Uvodni cast programu
G17 G40
G54 G90
'''
#Konečná část programu
Konecprg = ''' M30
'''
Komentar = ";" # Komentář pro Sinumerik
#----------------------------------------------------

#TOOLTIP-Napoveda při výběru postprocesoru v funkci JOB
TOOLTIP='''
Toto je postprocesor pro EMCO CONCEPT MILL 105
        '''

# otevření souboru
if open.__module__ == '__builtin__':
    pythonopen = open

#čísla řádku
def cislaradku():
    global Radek
    if CISLA_RADKU == True:
        Radek += 10 # hodnota o kterou se zvetsi následující cislo radku
        return "N" + str(Radek) + " "
```

```
#čísla řádku
def cislaradku():
    global Radek
    if CISLA_RADKU == True:
        Radek += 10 # hodnota o kterou se zvetsi následující cislo radku
        return "N" + str(Radek) + " "
    return ""


#Definování Exportní funkce - vstupni parametry seznam objektu Path modulu a nazev souboru
#a pripadne retezvoce argumenty (argstring)
def export(objectslist, filename,argstring):
    for obj in objectslist:
        if not hasattr(obj, "Path"):
            return None #Pokud objekt nema atrribut Path vrací prázdnou (None) hodnotu
    print("Probíha postprocessing...")
    vystup = ""
    # Hlavicka programu
    if Hlavicka:
        vystup += cislaradku() + ";Export souboru postprocesorem: " + __name__ + " \n"
        vystup += cislaradku() + ";Cas exportu: " + str(now.strftime('%d.%m.%Y %H:%M:%S')) + "\n"
    for line in Uvodprg.splitlines(True):
        vystup += cislaradku() + line
    for obj in objectslist:
        vystup += parse(obj)
    for line in Konecprg.splitlines(True):
        vystup += cislaradku() + line
        #Zobrazeni eidoru G-kodu před uložením soboru
    if FreeCAD.GuiUp:
        dia = PostUtils.GCodeEditorDialog()
        dia.editor.setText(vystup)
        result = dia.exec_()
        if result:
            final = dia.editor.toPlainText()
        else:
            final = vystup
    print("Postprocessing hotov.")
    #Otevření okno pro uložení
    gfile = pythonopen(filename,"w")
    gfile.write(final)
    gfile.close()

def parse(pathobj):
    global MODAL
    out = ""
    posledni_prikaz = None
    parametry = ['X', 'Y', 'Z', 'A', 'B', 'I', 'J', 'F', 'S', 'Q', 'R', 'L', 'H','T']#seznam prama
    modal_parametry = ['X', 'Y', 'Z','A', 'B', 'F', 'S', 'T']
    modalparam = {}
    prvnihodnota = Path.Command ("G1" {"X":-1 "Y":-1 "Z":-1 "F":0.0 "T":0})# nastaveni prvni
```

emco_post.py - C:\Users\Martin\Desktop\FreeCAD_0.17.12570_x64_dev_win\Mod\Path\PathScripts\post\emco_post.py (3.6.4)

File Edit Format Run Options Window Help

```
def parse(pathobj):
    global MODAL
    out = ""
    posledni_prikaz = None
    parametry = ['X', 'Y', 'Z', 'A', 'B', 'I', 'J', 'F', 'S', 'Q', 'R', 'L', 'H','T']
    modal_parametry = ['X', 'Y', 'Z','A', 'B', 'F', 'S', 'T']
    modalparam = {}
    prvnihodnota = Path.Command ("G1", {"X":-1, "Y":-1, "Z":-1, "F":0.0, "T":0})
    modalparam.update(prvnihodnota.Parameters)

    if hasattr(pathobj, "Group"):  # parsovani skupiny drah
        for p in pathobj.Group:
            out += parse(p)
        return out
    else:
        for c in pathobj.Path.Commands:
            outstring = []
            command = c.Name #Jmena jednotlivých příkazu z Command obj G,M,Komentar
            if command[0]=='(': #Koemntař ;
                command = PostUtils.fcoms(command, Komentar)
            outstring.append(command) # Přidaní příkazu
            if MODAL is True: #Neopakuje stejne priakzy (G,M)
                if command == posledni_prikaz:
                    outstring.pop(0)


            for param in parametry:
                if param in c.Parameters:
                    if param == 'F'and (modalparam[param] != c.Parameters[param]):
                        if c.Name not in ["G0", "G00"]:
                            speed = Units.Quantity(c.Parameters['F'], FreeCAD.Units.Velocity)
                            if speed.getValueAs('mm/min') > 0.0:
                                outstring.append(param + format(float(speed.getValueAs('mm/min')),'.0f' ))
                            else:
                                continue
                    elif param == 'I':
                        outstring.append(param + format(c.Parameters['I'], '.3f'))
                    elif param == 'J':
                        outstring.append(param + format(c.Parameters['J'], '.3f'))
                    elif param == 'T':
                            outstring.append(param + format(int(c.Parameters["T"])))
                    elif param == 'S':
                            outstring.append(param + format(int(c.Parameters[param])))
                    elif param =="Y" and (c.Name in ["G2", "G3"])and (modalparam["Z"] != c.Parameters["Z"]):
                            outstring.append(param + format(c.Parameters[param], '.3f'))
                    else:
                        if (param in modalparam) and (modalparam[param] == c.Parameters[param]):
                            continue
                        else:
                            outstring.append(param + format(c.Parameters[param], '.3f'))

            # ulozeni posledniho prikazu a modalniho parameteru
            posledni_prikaz = command
            modalparam.update(c.Parameters)

            # Prida retezec vymnena nastroje
            if command == 'M6':
                    out += cislaradku() + ";Vymena nastroje\n"

            # Cislo bloku jako prvni hodnota
            if len(outstring) >= 1:
                if CISLA_RADKU:
                    outstring.insert(0, (cislaradku()))

            # Pridani radku do finalniho vystupu
                for w in outstring:
                    out += w + " "
                out = out.strip() + "\n"

        return out


print(__name__ + " postprocessor načtený.")
```

PŘÍLOHA č. 2

Ukázka CL dat

Součást č.1

```
(This output produced with the dump post processor)
(Dump is useful for inspecting the raw commands in your paths)
(but is not useful for driving machines.)
(Consider setting a default postprocessor in your project or )
(exporting your paths using a specific post that matches your machine)

(Path: T1: D16)
Command (T1: D16) [ ]
Command M6 [ T:1 ]
Command M3 [ S:2000 ]
(Path: Pocket_Shape)
Command (Pocket_Shape) [ ]
Command G0 [ Z:20 ]
Command G0 [ X:6 Y:-20 ]
Command G0 [ Z:12.5 ]
Command G1 [ F:3.3333 X:7 Y:-7 Z:12.5 ]
Command G1 [ F:3.3333 X:7 Y:67 Z:12.5 ]
Command G1 [ F:3.3333 X:83 Y:67 Z:12.5 ]
Command G1 [ F:3.3333 X:83 Y:-7 Z:12.5 ]
Command G1 [ F:3.3333 X:7 Y:-7 Z:12.5 ]
Command G0 [ X:7 Y:-7 Z:20 ]
Command G0 [ X:18.851 Y:4.5387 Z:20 ]
Command G1 [ F:3.3333 X:18.851 Y:4.5387 Z:12.5 ]
Command G2 [ F:3.3333 I:26.149 J:25.461 K:0 X:46.211 Y:66.477 Z:12.5 ]
Command G2 [ F:3.3333 I:-1.2111 J:-36.477 K:0 X:81.477 Y:28.789 Z:12.5 ]
Command G2 [ F:3.3333 I:-36.477 J:1.2111 K:0 X:43.789 Y:-6.4775 Z:12.5 ]
Command G2 [ F:3.3333 I:1.2111 J:36.477 K:0 X:18.851 Y:4.5387 Z:12.5 ]
Command G0 [ Z:20 ]
...
(Path: T4: D8)
Command (T4: D8) [ ]
Command M6 [ T:4 ]
Command M3 [ S:2000 ]
(Path: Helix)
Command (Helix) [ ]
Command (helix cut operation) [ ]
Command G0 [ F:0 Z:20 ]
Command G0 [ F:0 X:49 Y:30 ]
Command G0 [ F:0 Z:18 ]
Command G1 [ F:3.3333 Z:15 ]
Command G2 [ F:3.3333 I:-4 J:0 X:41 Y:30 Z:14.583 ]
Command G2 [ F:3.3333 I:4 J:0 X:49 Y:30 Z:14.167 ]
Command G2 [ F:3.3333 I:-4 J:0 X:41 Y:30 Z:13.75 ]
Command G2 [ F:3.3333 I:4 J:0 X:49 Y:30 Z:13.333 ]
Command G2 [ F:3.3333 I:-4 J:0 X:41 Y:30 Z:12.917 ]
Command G2 [ F:3.3333 I:4 J:0 X:49 Y:30 Z:12.5 ]
Command G2 [ F:3.3333 I:-4 J:0 X:41 Y:30 Z:12.5 ]
Command G2 [ F:3.3333 I:4 J:0 X:49 Y:30 Z:12.5 ]
Command G0 [ F:0 Z:18 ]
Command G0 [ Z:20 ]
...
(Path: T6: D3)
Command (T6: D3) [ ]
Command M6 [ T:6 ]
Command M3 [ S:2000 ]
```

```
(Path: Pocket_Shape001)
Command (Pocket_Shape001) [ ]
Command G0 [ Z:20 ]
Command G0 [ X:54.899 Y:39.899 ]
Command G0 [ Z:18 ]
Command G1 [ F:3.3333 X:54.899 Y:39.899 Z:14 ]
Command G2 [ F:3.3333 I:-9.8988 J:-9.8988 K:0 X:45.999 Y:16.037 Z:14 ]
Command G2 [ F:3.3333 I:-0.99868 J:13.963 K:0 X:44.001 Y:43.963 Z:14 ]
Command G2 [ F:3.3333 I:0.99868 J:-13.963 K:0 X:54.899 Y:39.899 Z:14 ]
Command G0 [ X:54.899 Y:39.899 Z:20 ]
Command G0 [ X:54.192 Y:39.192 Z:20 ]
Command G1 [ F:3.3333 X:54.192 Y:39.192 Z:14 ]
Command G2 [ F:3.3333 I:-9.1916 J:-9.1916 K:0 X:43.865 Y:17.051 Z:14 ]
Command G2 [ F:3.3333 I:1.1352 J:12.949 K:0 X:46.135 Y:42.949 Z:14 ]
Command G2 [ F:3.3333 I:-1.1352 J:-12.949 K:0 X:54.192 Y:39.192 Z:14 ]
Command G1 [ F:3.3333 X:54.192 Y:39.192 Z:13 ]
Command G2 [ F:3.3333 I:-9.1916 J:-9.1916 K:0 X:43.865 Y:17.051 Z:13 ]
Command G2 [ F:3.3333 I:1.1352 J:12.949 K:0 X:46.135 Y:42.949 Z:13 ]
Command G2 [ F:3.3333 I:-1.1352 J:-12.949 K:0 X:54.192 Y:39.192 Z:13 ]
Command G0 [ X:54.192 Y:39.192 Z:20 ]
Command G0 [ X:54.899 Y:39.899 Z:20 ]
Command G1 [ F:3.3333 X:54.899 Y:39.899 Z:13 ]
Command G2 [ F:3.3333 I:-9.8988 J:-9.8988 K:0 X:45.999 Y:16.037 Z:13 ]
Command G2 [ F:3.3333 I:-0.99868 J:13.963 K:0 X:44.001 Y:43.963 Z:13 ]
Command G2 [ F:3.3333 I:0.99868 J:-13.963 K:0 X:54.899 Y:39.899 Z:13 ]
Command G0 [ Z:20 ]
```

PŘÍLOHA č. 3

Převedený ISO kód (ukázka)

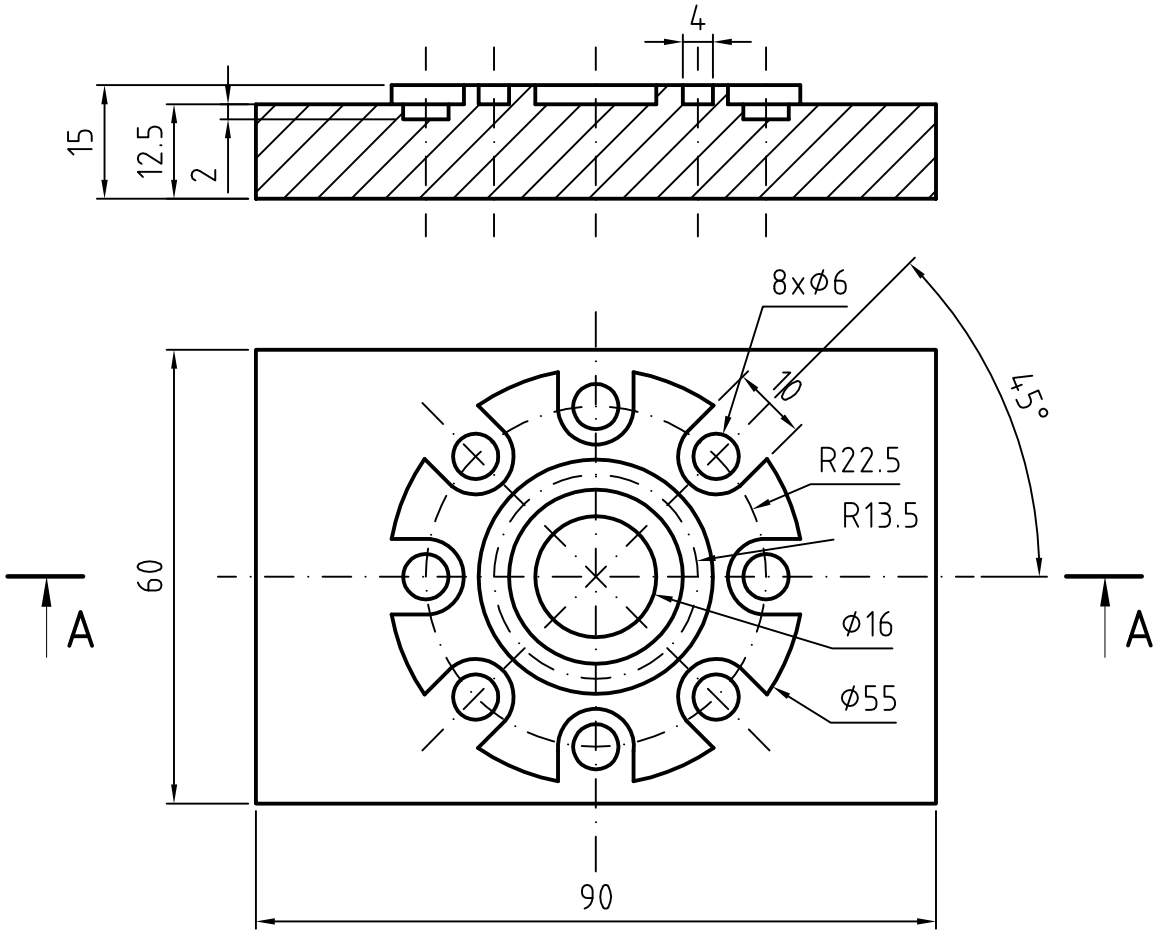
Součást č.1

```
N10 ;Export souboru postproceosrem: emco_post
N20 ;Cas exportu: 15.05.2018 21:09:46
N30 ; Uvodni cast programu
N40 G17 G40
N50 G54 G90
N60 ;T1: D16
N70 ;Vymena nastroje
N80 M6 T1
N90 M3 S2000
N100 ;Pocket_Shape
N110 G0 Z20.000
N120 X6.000 Y-20.000
N130 Z12.500
N140 G1 X7.000 Y-7.000 Z12.500 F200
N150 Y67.000
N160 X83.000
N170 Y-7.000
N180 X7.000
N190 G0 Z20.000
N200 X18.851 Y4.539
N210 G1 Z12.500
N220 G2 X46.211 Y66.477 I26.149 J25.461
N230 X81.477 Y28.789 I-1.211 J-36.477
N240 X43.789 Y-6.477 I-36.477 J1.211
N250 X18.851 Y4.539 I1.211 J36.477
N260 G0 Z20.000
...

N950 ;T4: D8
N960 ;Vymena nastroje
N970 M6 T4
N980 M3 S2000
N990 ;Helix
N1000 ;helix cut operation
N1010 G0 Z20.000
N1020 X49.000 Y30.000
N1030 Z18.000
N1040 G1 Z15.000 F200
N1050 G2 X41.000 Y30.000 Z14.583 I-4.000 J0.000
N1060 X49.000 Y30.000 Z14.167 I4.000 J0.000
N1070 X41.000 Y30.000 Z13.750 I-4.000 J0.000
N1080 X49.000 Y30.000 Z13.333 I4.000 J0.000
N1090 X41.000 Y30.000 Z12.917 I-4.000 J0.000
N1100 X49.000 Y30.000 Z12.500 I4.000 J0.000
N1110 X41.000 I-4.000 J0.000
N1120 X49.000 I4.000 J0.000
N1130 G0 Z18.000
N1140 Z20.000
...
N1960 ;T6: D3
N1970 ;Vymena nastroje
N1980 M6 T6
N1990 M3 S2000
N2000 ;Pocket_Shape001
N2010 G0 Z20.000
N2020 X54.899 Y39.899
N2030 Z18.000
```

```
N2040 G1 Z14.000 F200
N2050 G2 X45.999 Y16.037 I-9.899 J-9.899
N2060 X44.001 Y43.963 I-0.999 J13.963
N2070 X54.899 Y39.899 I0.999 J-13.963
N2080 G0 Z20.000
N2090 X54.192 Y39.192
N2100 G1 Z14.000
N2110 G2 X43.865 Y17.051 I-9.192 J-9.192
N2120 X46.135 Y42.949 I1.135 J12.949
N2130 X54.192 Y39.192 I-1.135 J-12.949
N2140 G1 Z13.000
N2150 G2 X43.865 Y17.051 I-9.192 J-9.192
N2160 X46.135 Y42.949 I1.135 J12.949
N2170 X54.192 Y39.192 I-1.135 J-12.949
N2180 G0 Z20.000
N2190 X54.899 Y39.899
N2200 G1 Z13.000
N2210 G2 X45.999 Y16.037 I-9.899 J-9.899
N2220 X44.001 Y43.963 I-0.999 J13.963
N2230 X54.899 Y39.899 I0.999 J-13.963
N2240 G0 Z20.000
N2250 M30
```

6,3

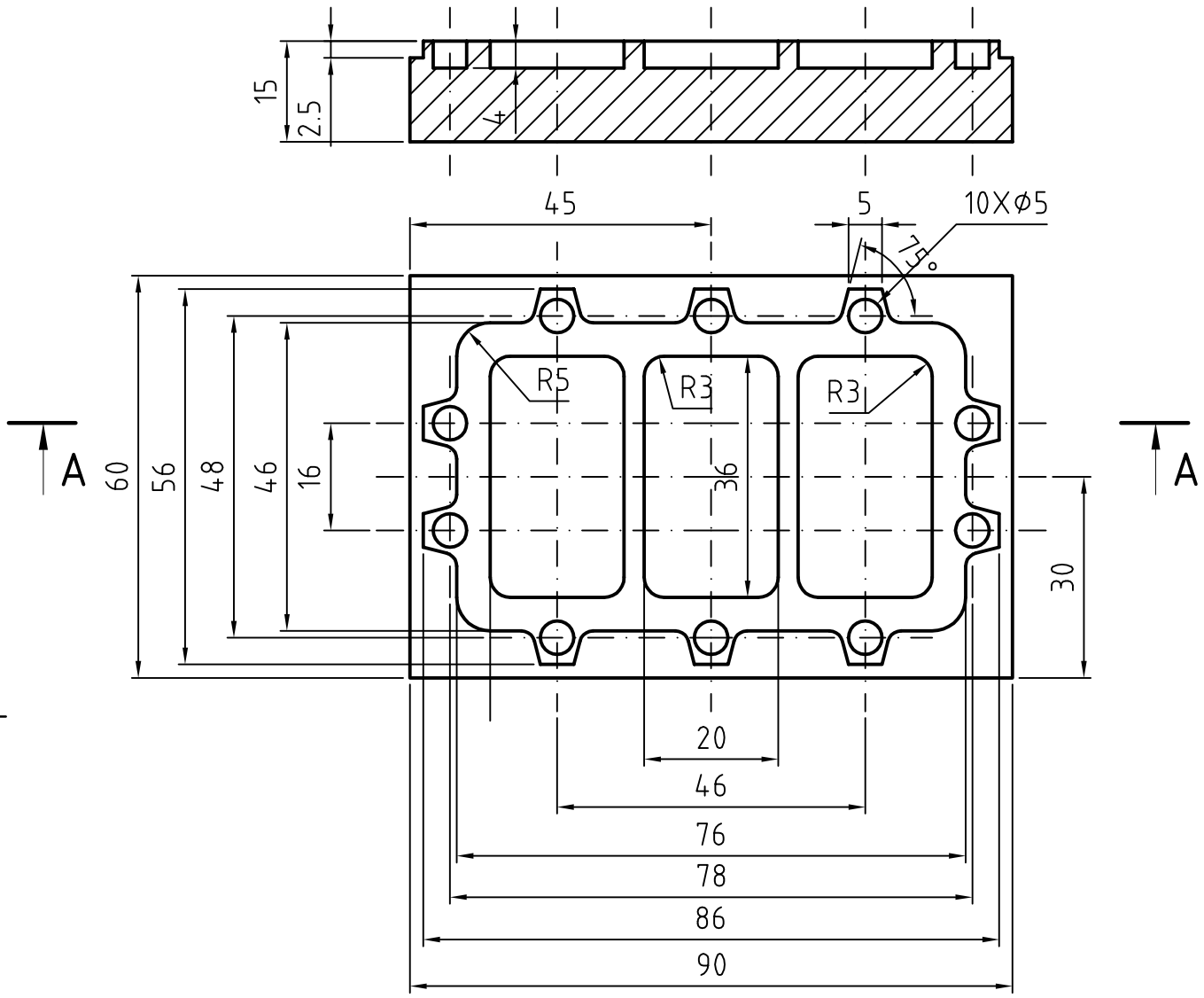
A-A



INDEX	ZMĚNA	DATUM	PODPIS	ZČU Plzeň	
ZN.MAT. PA 4.6			T.O.	HMOTNOST kg	MĚŘ. 1:1
ROZM.-POLOT. 90x60-15					
POM. ZAŘ.				ČSN	TŘ.Č.
VYPR. Vintr	NORM.REF.			POZN.	Č. KUSOVNÍKU
PŘEZK.					
TECHNOL.	SCHVÁLIL 20.5.2010			STARÝ V.	Č.V.
NÁZEV SOUČÁST Č.1				BP KTO 10-1	
				Listů	List

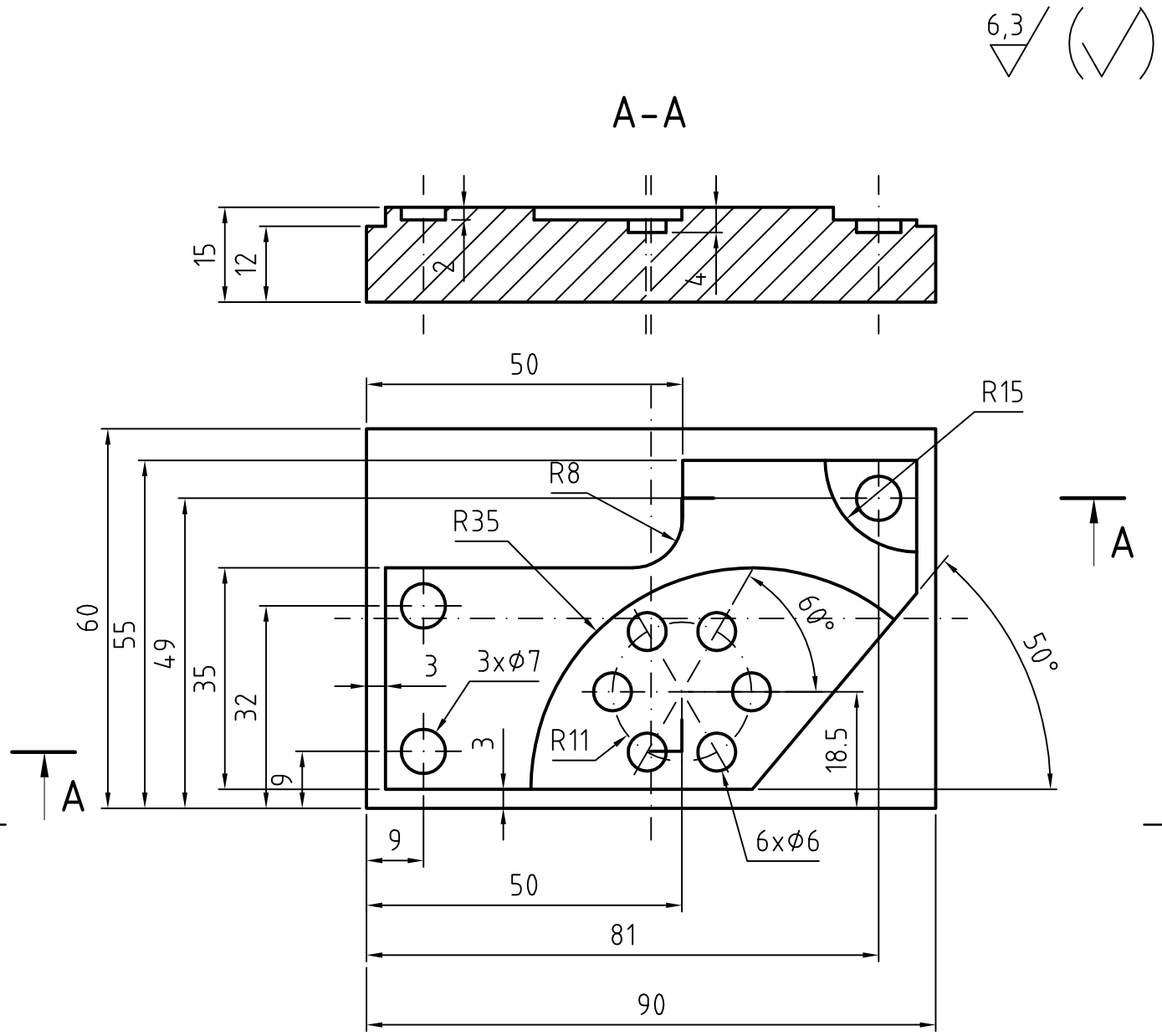
6,3

A-A



Nekótované rádiusy R2

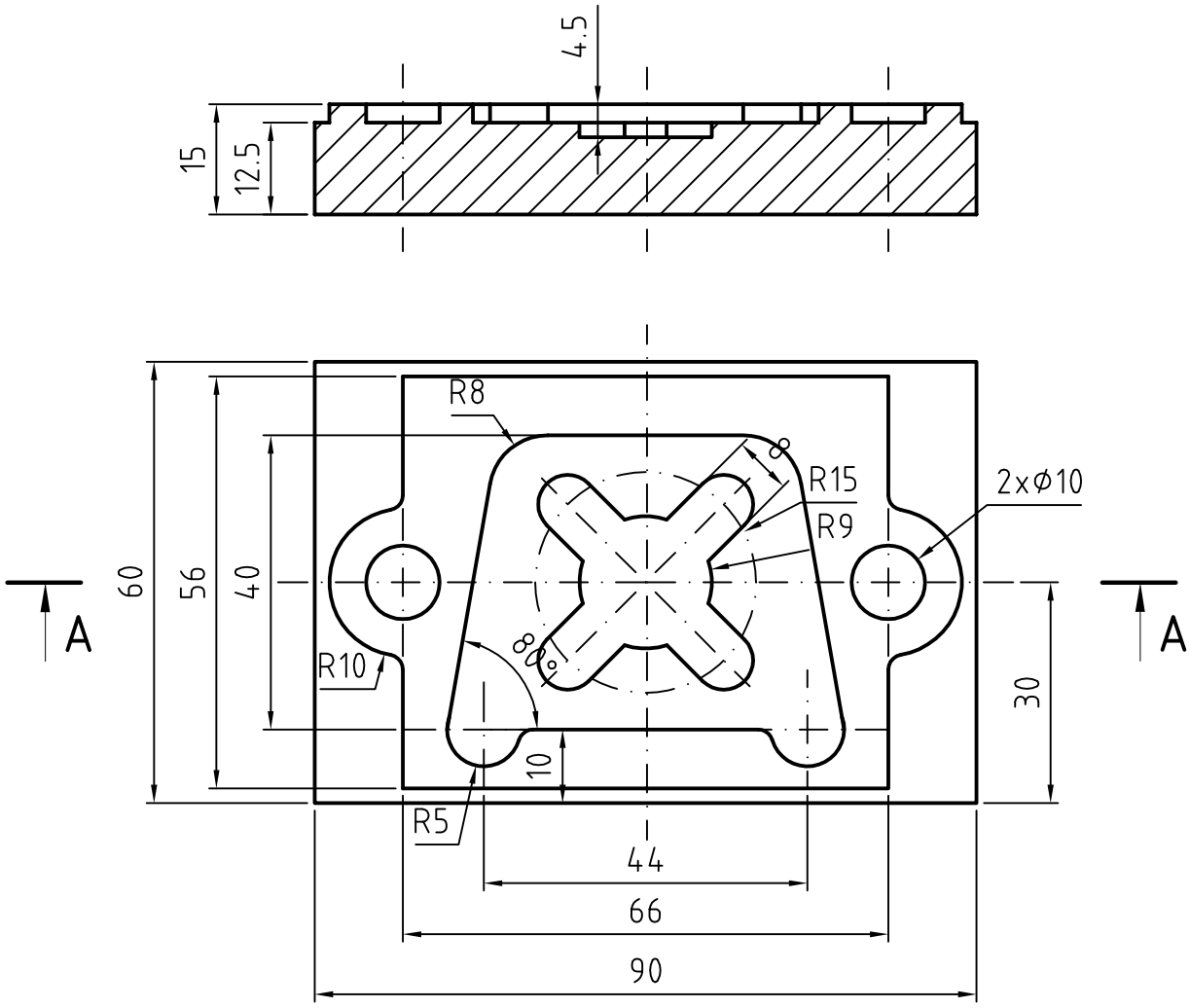
INDEX	ZMĚNA	DATUM	PODPIS	ZČU Plzeň	
ZN.MAT. PA 4.6			T.O.	HMOTNOST kg	MĚŘ. 1:1
ROZM.-POLOT. 90x60-15					
POM. ZAŘ.				ČSN	TŘ.Č.
VYPR. Vintr	NORM.REF.			POZN.	Č. KUSOVNÍKU
PŘEZK.					
TECHNOL.	SCHVÁLIL 20.5.2010			STARÝ V.	Č.V.
NÁZEV SOUČÁST Č.2				BP KTO 10-2	
				Listů	List



INDEX	ZMĚNA	DATUM	PODPIS	ZČU Plzeň	
ZN.MAT. PA 4.6			T.O.	HMOTNOST kg	MĚŘ. 1:1
ROZM.-POLOT. 90x60-15					
POM. ZAŘ.				ČSN	TŘ.Č.
VYPR. Vintr	NORM.REF.			POZN.	Č. KUSOVNÍKU
PŘEZK.					
TECHNOL.	SCHVÁLIL 20.5.2010			STARÝ V.	Č.V.
NÁZEV SOUČÁST Č.18				BP KTO 10-18	
				Listů	List

6,3

A-A



Nekótované rádiusy R2

INDEX	ZMĚNA	DATUM	PODPIS	ZČU Plzeň	
ZN.MAT. **PA 4.6**			T.O.	HMOTNOST kg	MĚŘ. 1:1
ROZM.-POLOT. **90x60-15**					
POM. ZAŘ.				ČSN	TŘ.Č.
VYPR. **Vintr**	NORM.REF.			POZN.	Č. KUSOVNÍKU
PŘEZK.					
TECHNOL.	SCHVÁLIL **20.5.2010**			STARÝ V.	Č.V.
NÁZEV SOUČÁST Č.19				BP KTO 10-19	
				Listů	List