ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA STROJNÍ

KATEDRA TECHNOLOGIE OBRÁBĚNÍ

Studijní program: **N2301 Strojní inženýrství**

Studijní zaměření: **Strojírenská technologie – technologie obrábění a aditivní technologie**

DIPLOMOVÁ PRÁCE

Tvorba externí aplikace pro generování seřizovacího listu

Autor: **Bc. Václav Touš**

Vedoucí práce: **Ing. Aneta Milsimerová Ph.D.**

Akademický rok 2020/2021

Originál (kopie) zadání DP

Katedra technologie obrábění Václav Touš

ANOTAČNÍ LIST DIPLOMOVÉ PRÁCE

AUTOR	**Příjmení** Touš	**Jméno** Václav	
STUDIJNÍ OBOR	Technologie obrábění		
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Ing. Milsimerová Ph.D.	**Jméno** Aneta	
PRACOVIŠTĚ	ZČU – FST – KTO		
DRUH PRÁCE	**DIPLOMOVÁ**	~~**BAKALÁŘSKÁ**~~	**Nehodící se škrtněte**
NÁZEV PRÁCE	Tvorba externí aplikace pro generování seřizovacího listu		

FAKULTA	strojní		**KATEDRA**	KTO	**ROK ODEVZD.**	2021

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	68		**TEXTOVÁ ČÁST**	59	**GRAFICKÁ ČÁST**	8

STRUČNÝ POPIS (MAX 10 ŘÁDEK) ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY	Diplomová práce se zabývá tvorbou aplikace pro automatizované generování seřizovacího listu. Konkrétně jsou zmapovány možnosti tvorby seřizovacího listu v systému InventorHSMpro. V tomto systému je tvořen postprocesor pro výstup tohoto listu v programovacím jazyce JavaScript. Samotná aplikace je tvořena v prostředí VisualStudio a MicrosoftWindows pomocí programovacího jazyka C#. Aplikace je plně implementována do prostředí InventorHSM.
KLÍČOVÁ SLOVA	Seřizovací list, InventorHSM, C#, JavaScript, API, VisualStudio, DATA SHEET TOOL MANAGER, postprocesoring, proměnná, funkce, script, aplikace, integrace.

Katedra technologie obrábění Václav Touš

SUMMARY OF DIPLOMA SHEET

AUTHOR	**Surname** Touš	**Name** Václav	
FIELD OF STUDY	Machining technology		
SUPERVISOR	**Surname (Inclusive of Degrees)** Ing. Milsimerová Ph.D.	**Name** Aneta	
INSTITUTION	ZČU – FST – KTO		
TYPE OF WORK	**DIPLOMA**	~~**BACHELOR**~~	**Delete when not applicable**
TITLE OF THE WORK	Creating an external application for generating a data sheet		

FACULTY	Mechanical Engineering		**DEPARTMENT**	KTO		**SUBMITTED IN**	2021

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	68		**TEXT PART**	58		**GRAPHICAL PART**	8

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	The diploma thesis deals with the creations of an application for the automated generation of an data sheet. Specifically, the possibilities of creating an adjustment sheet in the InventorHSM system are maped. In this system, a postprocessor is created for the output of this sheet in the JavaScript programming language. The application itself is created in VisualStudio and MicrosoftWindows using the C# programming language. The application is fully implemented in the InventorHSM.
KEY WORDS	Data sheet, InventorHSM, C#, JavaScript, API, DATA SHEET TOOL MANAGER, postprocessoring, variable, function, script, application, integration.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této práce.

Dále prohlašuji, že veškerý software použitý při řešení této diplomové práce je legální.

.....

V Plzni dne: 31.5.2021 Bc. Václav Touš

Poděkování

Rád bych poděkoval své vedoucí diplomové práce Ing. Anetě Milsimerové Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce. Děkuji také Ing. Ladislavu Peleškovi za to, že si na mě vždy udělal čas a byl mi schopen s čímkoliv poradit. Velké poděkování patří taktéž mým nejbližším, kteří mě při psaní mé práce podporovali.

Obsah

Seznam symbolů a zkratek

API	Application Programming Interface / Rozhraní pro programování aplikací
C#	C Sharp
CAD	Computer Aided design / Počítačem podporované projektování
CAE	Computer aided engineering / Počítačem podporované inženýrství
CAM	Computer Aided Manufacturing / Počítačem podporovaná výroba
CL data	Cutter location data / Data polohy referenčního bodu nástroje
CNC	Computer Numerical Control / Počítačové číslicové řízení
DNC	Direct NumericalControl/ Přímé číslicové řízení
HSM	High speed machining
JS	JavaScript
NC	Numerical Control / Číslicové řízení
PC	Počítač
PLC	Programmable logic control / Programovatelný logický automat
SW	Software
VS	Visual Studio

1 Úvod

1.1 Cíl této práce

Tento projekt je zaměřen na tvorbu vlastní externí aplikace, která bude sloužit pro automatické generování seřizovacího listu. Tato aplikace by měla původní seřizovací list od společnosti Autodesk obohatit o možnost vložení screenů pořízené během simulace obrábění, možnost přiřazení poznámky danému screenu a volbu formátu v jakém bude seřizovací list generován (_.pdf, _.xlsx, _.html). K těmto atributům bude automaticky přiřazen název operace ve které byl snímek pořízen, název použitého nástroje v dané operaci, popřípadě definice nulového bodu.

Velkou část této aplikace bude tvořit script původního seřizovacího listu, který byl dodán v základní verzi společností Autodesk a je součástí balíčku CAD/CAM. Tento script bude kompletně upraven a naformátován dle nových požadavků a potřeb uživatelů.

Vizuální stránka seřizovacího listu bude upravena tak, aby jasně a přehledně obsáhla a definovala veškeré potřebné informace, včetně obrázkových příloh. Důležitost těchto informací je stanovena ze znalosti technologie obrábění, potřebám informovanosti obsluhy a vzájemné komunikaci technolog–programátor–obsluha stroje. Finální verzi se všemi detaily lze individuálně optimalizovat dle přání jednotlivých uživatelů daného CAM SW. Návrh nové základní verze bude již optimalizován dle nejčastějších požadavků uživatelů.

Seřizovací list je tvořen v prostředí JavaScript, které je pomocí API rozhraní provázán s danou aplikací, která je vytvořena v prostředí Visual Studia pomocí jazyku C#.

Základní formát výstupu seřizovacího listu bude soubor _.pdf. Ve vytvořené aplikaci bude možnost volby generování seřizovacího listu ve formátech _.html a _.xlsx.

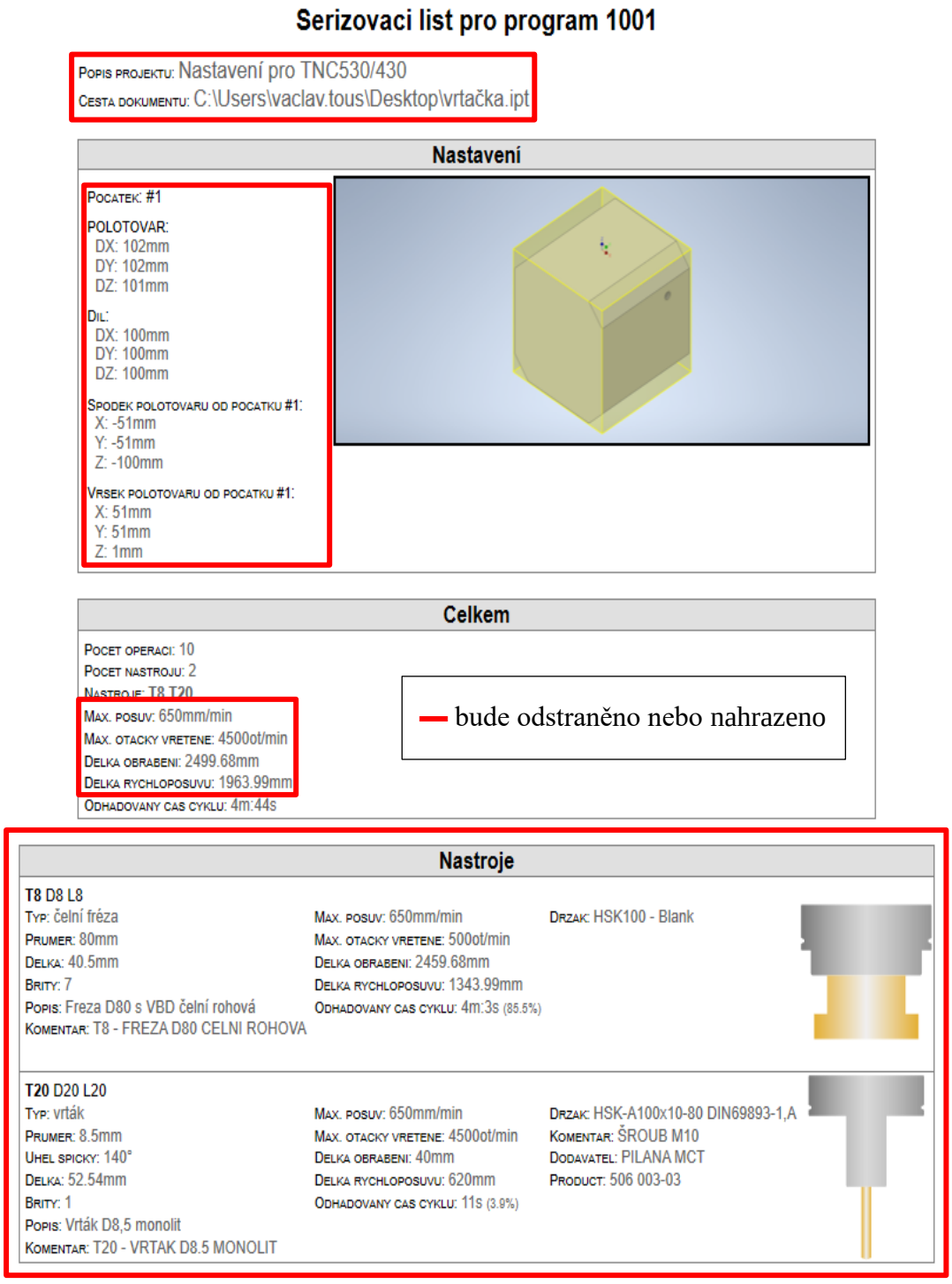
1.2 K čemu slouží seřizovací list

Seřizovací list slouží obsluze stroje jako manuál pro jeho správné nastavení a seřízení stroje k jednotlivým obráběcím operacím a cyklům. Specifikuje správné upnutí obrobku. Definuje umístění nulového bodu nebo jejich posunutí. Automaticky generuje seznam

nástrojů, které byly použity při programování v CAM SW. Jsou vypsány řezné podmínky u jednotlivých operací a nástrojů. Nástroje jsou zde popsány z hlediska jejich velikostí, tvaru a použití. Dále se dělí na jejich upnutí, potřebné vyložení, minimální délku ostří a další parametry, které jsou nedílnou součástí správného procesu při seřizování stroje, přípravě upínání obrobku a přípravě nástrojů.

Dalšími důležitými informacemi pro správné seřízení stroje jsou korekce nástrojů. Většinou se o těchto korekcích bavíme jako o „délkových“, nebo „průměrových“, tedy o kompenzaci rozměru nástroje v určeném směru vůči skutečné poloze obrobku. Jejich zápis je závislý na použitém řídicím systému. Zpravidla se do seřizovacího listu používá podobný, ne-li totožný zápis, jako při definici dané korekce přímo ve stroji nebo v programu.

2 Rozbor současného stavu



Obr. 1: Původní seřizovací list nástroje.

Operace

Operace 1/10 Popis: W +125;Celo B=0° Strategie: Celni Pocatek: #1 Tolerance: 0.01mm Max. krok dolu: 1mm Max. stranovy krok: 76mm	Max. otacky vretene: 500ot/min Max. posuv: 650mm/min Delka obrabeni: 665.98mm Delka rychloposuvu: 240.54mm Odhadovany cas cyklu: 1m:4s (22.6%) Chlazeni: Kapalina	**T8** D8 L8 Typ: čelní fréza Prumer: 80mm Delka: 40.5mm Brity: 7 Popis: Freza D80 s VBD čelní rohová Komentar: T8 - FREZA D80 CELNI ROHOVA	
Operace 2/10 Popis: W +125;Celo B=+45° Strategie: Celni Pocatek: #1 Tolerance: 0.01mm Max. stranovy krok: 76mm	Max. otacky vretene: 500ot/min Max. posuv: 650mm/min Delka obrabeni: 181.36mm Delka rychloposuvu: 148.02mm Odhadovany cas cyklu: 19s (6.5%) Chlazeni: Kapalina	**T8** D8 L8 Typ: čelní fréza Prumer: 80mm Delka: 40.5mm Brity: 7 Popis: Freza D80 s VBD čelní rohová Komentar: T8 - FREZA D80 CELNI ROHOVA	
Operace 3/10 Popis: W +140;Celo B=90° Strategie: Celni Pocatek: #1 Tolerance: 0.01mm Max. krok dolu: 1mm Max. stranovy krok: 76mm	Max. otacky vretene: 500ot/min Max. posuv: 650mm/min Delka obrabeni: 534.83mm Delka rychloposuvu: 239.78mm Odhadovany cas cyklu: 52s (18.4%) Chlazeni: Kapalina	**T8** D8 L8 Typ: čelní fréza Prumer: 80mm Delka: 40.5mm Brity: 7 Popis: Freza D80 s VBD čelní rohová Komentar: T8 - FREZA D80 CELNI ROHOVA	
Operace 4/10 Popis: W +140;Celo B=+135° Strategie: Celni Pocatek: #1 Tolerance: 0.01mm Max. stranovy krok: 76mm	Max. otacky vretene: 500ot/min Max. posuv: 650mm/min Delka obrabeni: 180.66mm Delka rychloposuvu: 147.31mm Odhadovany cas cyklu: 18s (6.5%) Chlazeni: Kapalina	**T8** D8 L8 Typ: čelní fréza Prumer: 80mm Delka: 40.5mm Brity: 7 Popis: Freza D80 s VBD čelní rohová Komentar: T8 - FREZA D80 CELNI ROHOVA	
Operace 5/10 Popis: W +180;Celo B=180° Strategie: Celni Pocatek: #1 Tolerance: 0.01mm Max. krok dolu: 1mm Max. stranovy krok: 76mm	Max. otacky vretene: 500ot/min Max. posuv: 650mm/min Delka obrabeni: 266.92mm Delka rychloposuvu: 136mm Odhadovany cas cyklu: 26s (9.2%) Chlazeni: Kapalina	**T8** D8 L8 Typ: čelní fréza Prumer: 80mm Delka: 40.5mm Brity: 7 Popis: Freza D80 s VBD čelní rohová Komentar: T8 - FREZA D80 CELNI ROHOVA	

Obr. 2: Původní seřizovací list operace.

Jak je vidět na Obr. 1 a Obr. 2, původní seřizovací list obsahuje informace, které jsou generovány z prostředí Inventor HSM.

Tento seřizovací list popisuje:

- použité nástroje,
- jednotlivé operace,
- popis polotovaru,
- grafické zobrazení obráběné součásti a polotovaru,
- grafické zobrazení nástroje a držáku,

- řezné podmínky,
- časy obrábění,
- jiné informace, které jsou automaticky vygenerovány dle požadovaného nastavení v aplikaci CAM.

Při podrobnějším rozboru seřizovacího listu lze konstatovat, že ne všechny informace, které jsou automaticky generovány pomocí původní verze od společnosti Autodesk, jsou při seřizování stroje opravdu nutné nebo přímo potřebné. Seřizovací list je samozřejmě upravován dle požadavků uživatele, avšak nová základní verze by měla obsahovat potřebné informace, které jsou pro správné seřízení stroje očekávatelné, ne-li přímo nutné. Předpoklad je tedy takový, že nynější informace budou buďto odstraněny nebo nahrazeny jinými. Musí být brán v potaz i grafický vzhled seřizovacího listu.

I po doplnění a úpravách by se měla zachovat jeho:

- úplnost,
- přehlednost,
- jednoznačnost generovaných hodnot, informací

Úpravy se budou týkat i zobrazení obráběného dílu a polotovaru, tedy jeho velikosti, umístění a popisu.

2.1 Možnosti nového seřizovacího listu

Možnosti rozšíření nového seřizovacího listu jsou omezeny hned několika aspekty. Nesmí se opomenout, jaká bude jeho finální forma a jak bude konečným uživatelem používán. I když máme možnost doplnit seřizovací list třeba o vizualizaci, video, stínování atd. předpokládá se, že drtivá většina uživatelů bude požadovat seřizovací list v papírové podobě. Z toho vyplývá, že veškeré informace, které chceme obsluze poskytnout, musí být přenositelné z PC na papír. Využití interaktivních simulací v seřizovacím listě lze použít za předpokladu, že obsluha u stroje má možnost stejného rozhraní ve kterém byl soubor vytvořen (externí PC, tablet, možnost načtení do stroje).

Dalším omezením je samotné rozhraní HSM API. Toto rozhraní disponuje knihovnou vytvořenou přímo pro Inventor HSM. Pokud v této knihovně není funkce nebo předmět který chceme používat, musí být externí aplikace programována ručně,

tedy bez využití těchto již předem definovaných a upravených funkcí, které ve své podstatě jen vyvoláváme a přemisťujeme dle potřeby. Ruční programování zde může být pro řešení složitějších funkcí značně problematické a časově náročné.

Největším omezením je SW jako takový. Nejen že jsme omezeni dostupnými funkcemi a příkazy v InventorHSM, ale i možnostmi jiného zaevidování příkazu pro další automatické generování.

2.2 Návrh nového seřizovacího listu

Nová verze seřizovacího listu je vždy navrhována, jak se říká „na míru“ zákazníkovi nebo jakémukoliv uživateli daného SW. Každý seřizovací list by měl být takto upraven a odlazen. Jelikož původní verze je natolik rozdílná od požadovaného výstupu, bude vytvořena nová základní verze. Informace, kterými bude disponovat, byly vybrány podle nejčastěji požadovaných výstupů od samotných uživatelů. Tyto požadavky byly nejen na jejich rozšíření o potřebná data, která byla pro správné seřízení nutná, ale i na eliminaci nepotřebných informací. Tyto nepotřebné informace ve výsledném, vygenerovaném seřizovacím listě zabírají svým textem nadbytečné množství místa a seřizovací list je tak méně přehledný. Když se bavíme o nepotřebných informacích, měli bychom zde uvést příklad, jelikož požadavky zákazníků jsou různé. I přesto byla tato data zařazena mezi zbytečná, alespoň tedy pro obsluhu.

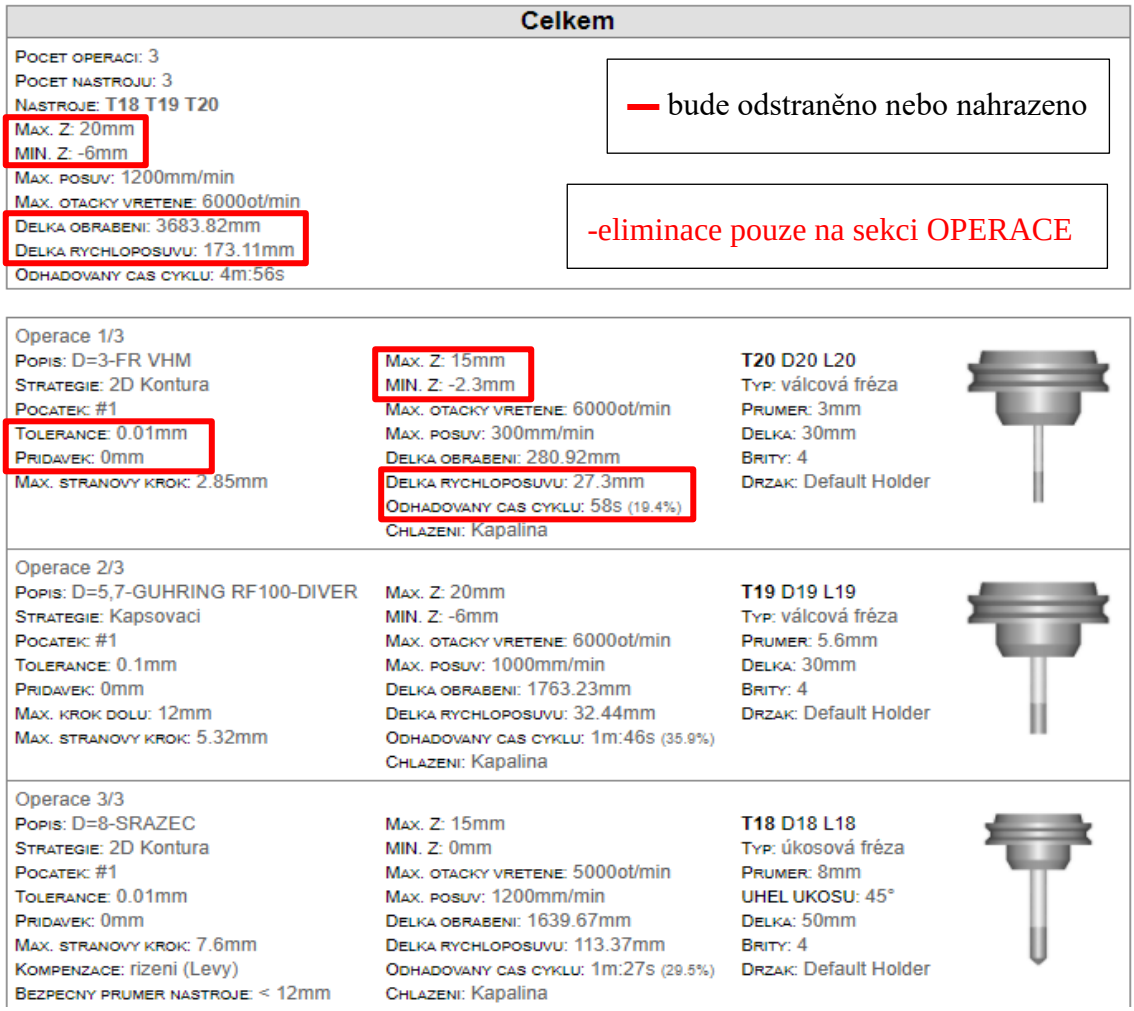
Příklady nadbytečných informací:

- délka obrábění-je tím myšlena délka veškerých operací v mm,
- délka rychloposuvu-délka všech zrychlených posuvů v mm,
- posunutí polotovaru vůči původnímu souřadného systému – nejedná se zde o volbu nulového bodu nebo jeho posunutí, ale o posunutí modelu vůči souřadnému systému při tvorbě jeho náčrtu.

Jak je vidět na Obr. 3, informace jsou rozděleny do tří hlavních sekcí:

- hlavička,
- operace,
- nástroj.

Sekce operace a nástroj budou sloučeny do jedné. Množství informací, které jsou v původní verzi nejsou žádoucí a některé z nich se i opakují. Z hlediska potřebných informací není nutné rozdělovat seřizovací list do více sekcí než dvou. První bude hlavička seřizovacího listu, kde budou uvedeny základní informace o projektu. Druhá sekce OPERACE bude obsahovat informace jak o operaci, tak o nástroji a držáku. Pro operace i nástroj bude tento prostor naprosto postačující, aniž by bylo nutné eliminovat důležité informace pro obsluhu. Náhled si můžeme prohlédnout na Obr. 3, kde vidíme rozdíl v uskupení jednotlivých částí oproti původnímu seřizovacímu listu. Tento náhled slouží pouze pro představu rozložení jednotlivých parametrů, jejich hodnoty budou postupně upravovány.



Obr. 3: Rozložení nového seřizovacího listu.

2.3 Návrh externí aplikace

Aplikace by měla sloužit pro automatické generování seřizovacího listu z prostředí CAM SW InventorHSM. Tato aplikace spustí soubor _.cps, který obsahuje veškeré náležitosti seřizovacího listu, který byl graficky a obsahově upraven dle potřeb uživatele.

Tento soubor (respektive finální výstup) bude obsahovat přidané informace a vizuální zobrazení, která byla nahrána a zanesena pomocí vytvořené aplikace. Jde o možnost přidání screenů během simulace, přidání poznámky k jednotlivým screenům a možnost generování v požadovaném formátu (Obr.17).

Tyto funkce budou zobrazeny v grafickém okně externí aplikace. Pro vytvořené obrázky bude možnost náhledu a k jednotlivým obrázkům budou přiřazeny formuláře pro vytvoření poznámky.

2.4 Rozšíření nového seřizovacího listu pomocí externí aplikace

Nová externí aplikace integrována do prostředí InventorHSM doplní seřizovací list o obrázky, které byly pořízeny během simulace v prostředí CAM SW. Bude zde možnost volby formátu, ve kterém má být seřizovací list generován.

Dále bude možnost obrázky doplnit o poznámky uživatele ve formě textu. Každá poznámka bude uvedena u příslušného screenu v prostředí aplikace. Tento text bude automaticky přiřazen k obrázku v generovaném listě na příslušném místě a podle nastavení grafického okna. Tyto funkce budou hlavním pilířem pro tvorbu nové aplikace.

Funkce aplikace:

- ScreenShot 1, ScreenShot 2, ScreenShot 3;
- Komentář 1, Komentář 2, Komentář 3;
- Formát: _.xls, _.pdf, _.html;
- Skrýt dialogové okno;
- Generovat seřizovací list;
- Otevřít složku;
- Konec programu.

2.5 Vlastní vývoj aplikace

Do budoucna je zde možnost rozšířit aplikaci o další funkce. Lze pracovat i s modelovou částí aplikace Inventor, například při práci se sestavami, jejich nastavení pro obrábění a následné vizualizace. Do těchto sestav (popřípadě podsestav) je upevněn díl, který má být obráběn. Jelikož knihovna APIHSM má možnost napojení na parametrické modelování,

lze pomocí nich velice rychle nadefinovat například upínky pro danou sestavu. Tyto upínky jsou modelovány v klasickém prostředí Inventoru, tudíž je možnost vytvořit jakýkoliv požadovaný tvar.

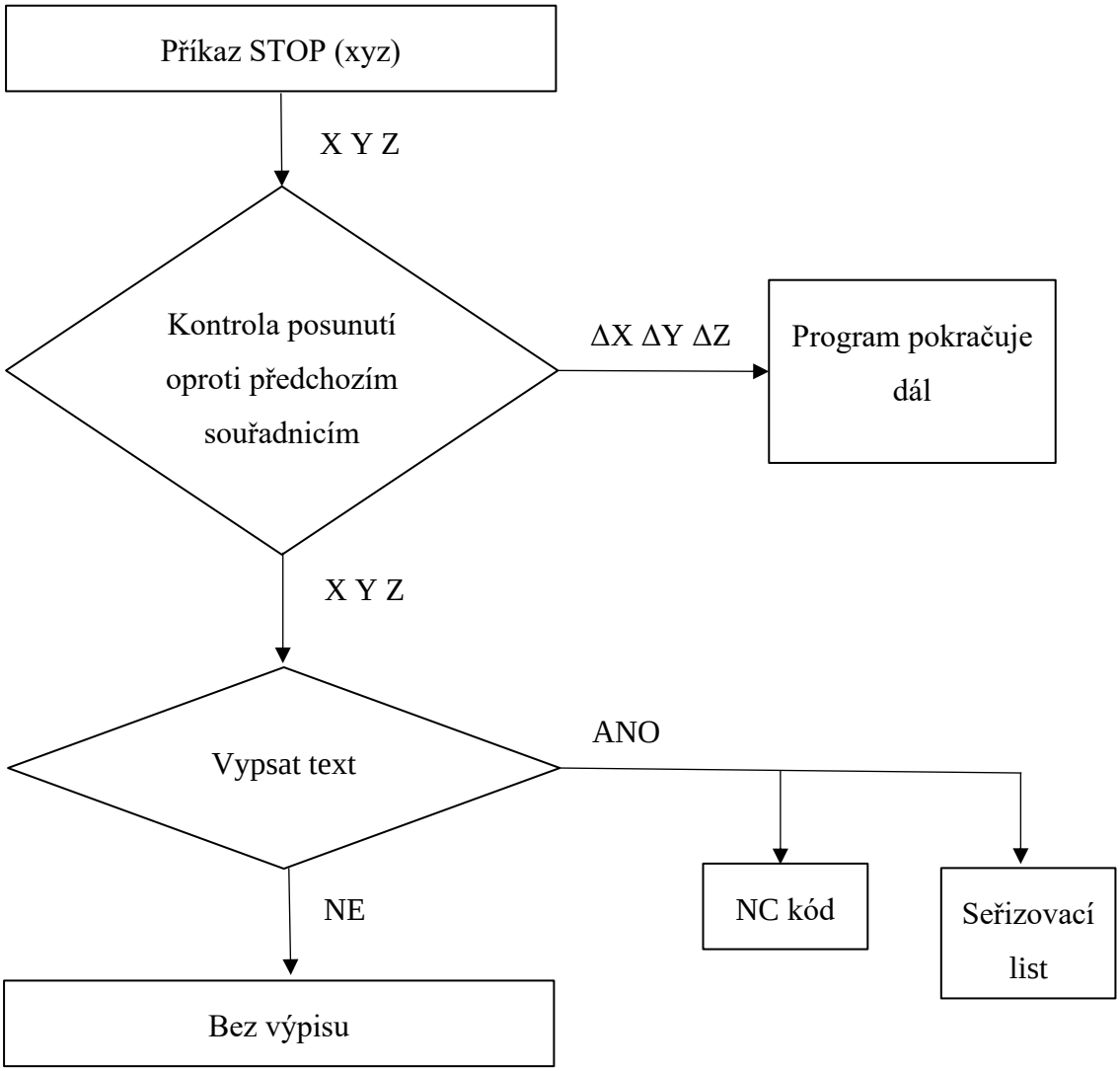
Dále je zde možnost přiřazení dalších parametrů k nástrojům a jejich držákům, operacím a cyklu samotnému, například samostatné komentáře, obrázky, fotky nebo jakékoliv pomocné modely.

Všechny tyto informace, parametry a data lze mezi sebou různě kombinovat. Vlastní aplikace HSM, na ně navázané knihovny API a vlastní programování (Obr.17).

Pokud se bavíme o hodnotách jednotlivých parametrů, ať jsou to rozměry modelů, sestav, nástrojů, ale i na ně navázané vygenerované dráhy obrábění, veškeré tyto hodnoty jsou generovány výpočtovým jádrem InventorHSM. Zde je aplikována synchronizace původně dvou výpočtových jader Inventor a HSM. Tyto hodnoty nazýváme CL data. CL data lze dále modifikovat, transformovat a formátovat dle vlastních potřeb programování. CL data nám udávají hodnotu. Až tato hodnota může být převedena na výraz nebo nenumerický parametr.

Příklad:

V CAM SW zadám příkaz STOP během tvorby programu pro obrábění. Výpočtové jádro dostane informaci o zastavení obrábění nebo stroje. Tato informace znamená nulové posunutí vůči předešlým souřadnicím. Zkontrolování podmínky výpočtovým jádrem. Vygenerování výrazu. Do NC kódu to může znamenat M00, nebo M01. Do seřizovacího listu poznámku STOP, zastavení stroje atd., dle nastavení postprocesoringu (Obr. 4).



Obr. 4: Vývojový diagram generování funkce STOP.

Poznámka:

Datové soubory polohování (CL) jsou generovány z cest obrábění specifikovaných v sekvencích NC. Každá NC sekvence generuje samostatný soubor CL. Lze také vytvořit jediný soubor pro celou operaci nebo cyklus (Obr. 5)

```
346: onCycle()
  cycleType='chip-breaking'
  cycle.clearance=5
  cycle.retract=5
  cycle.stock=0
  cycle.depth=91.35623931884766
  cycle.feedrate=80
  cycle.retractFeedrate=1000
  cycle.plungeFeedrate=80
  cycle.dwell=0
  cycle.incrementalDepth=2.5
  cycle.incrementalDepthReduction=0
  cycle.minimumIncrementalDepth=2.5
  cycle.accumulatedDepth=36.25
  cycle.chipBreakDistance=0.28999999165534973
  cycle.bottom=-91.35623931884766
  cycle.plungesPerRetract=14
  cycle.shift=0
  cycle.shiftOrientation=0
  cycle.compensatedShiftOrientation=0
  cycle.shiftDirection=3.141592653589793
  cycle.backBoreDistance=0
346: onCyclePoint(-31, 0, -91.35623931884766)
347: onCyclePoint(31, 0, -91.35623931884766)
348: onCycleEnd()
```

Obr. 5: Ukázka generování cyklu z výpočtového jádra.

2.6 Uspořádání scriptu seřizovacího listu

Script seřizovacího listu je psán ve stejném jazyce, ve kterém je modifikován postprocesoring pro InventorHSM (tedy v jazyce JavaScript). Script seřizovacího listu funguje na stejném principu, jako scripty těchto posprocesorů. Jelikože se bavíme o jazyce JS je nutné zmínit, že tento jazyk je oproti většině jiných komerčních programovacích jazyků asynchronní. I proto je používán právě pro tyto úpravy, a nejen například pro tvorbu webových stránek a jejich vývoj.

Poznámka:

U synchronního kódu se čeká na jeho dokončení, než se provede další akce (Obr. 6). U asynchronního kódu se na jeho dokončení nečeká a pokračuje se dál ve výpočtu (Obr. 7).

```
console.log(1);
setTimeout(() => console.log(2), 100); // po zavolání setTimeout se hned provede další kód
console.log(3);
```

Obr. 6: Synchronní programování.

Čeká se na výsledek první operace (sum), než se provede další operace.

```
const result = sum(4, 5);
console.log(result);
```

Obr. 7: Asynchronní programování.

Jako první se provede console.log(1). Poté se zavolá funkce setTimeout(...) a ihned se provede další operace console.log(3). Po 100 milisekundách se provede anonymní funkce, kterou jsme předali jako první parametr funkce setTimeout.

Při asynchronním zpracování je obvykle potřeba reagovat na dokončení asynchronního zpracování (například dál pracovat s jeho výsledkem) a k tomu JavaScript využívá tzv. callback funkce. Princip založený na funkcionálním programování, kdy funkce může být proměnnou, a tak může být použita jako parametr nebo návratová hodnota jiné funkce.

Script seřizovacího listu je rozdělen do několika tříd. Zde jsou ty nejzákladnější, které najdeme ve většině těchto kódů. Každý z nich má svoji specifickou funkci (Příloha 1):

- loadSharedStrings();
- saveSharedStrings();
- dumpIds();
- loadXML();
- formatTime();
- onSectionEnd();

- getStrategy();
- replaceVariables();
- updateWorkbook();
- onClose();
- onTerminate().

2.7 Formát seřizovacího listu

Výstupní formáty seřizovacího listu je možné generovat ve dvou základních variantách, které verze InventorHSM nabízí. Jde o soubory _.xls, nebo _.html. Tyto soubory lze modifikovat dle potřeb uživatele, ať jde o grafické úpravy, nebo daný obsah, který je automaticky generovaný z prostředí CAM SW.

Jedním z cílů této práce je možnost převedení těchto formátů do _.pdf souboru. Tento převod by neměl být automatický. Možnost volby formátu bude zohledněna ve vytvořené aplikaci.

DATA SHEET TOOL MANAGER-InventorHSMpro 2021	
HLAVIČKA vygenerovaných příloh	
	Poznámka_1
OBRÁZEK 1	
	Poznámka_2
OBRÁZEK 2	
	Poznámka_3
OBRÁZEK 3	

Obr. 8: Předpokládaný výstup seřizovacího listu s generovanými parametry.

3 Programovací prostředí

3.1 Prostředí JavaScriptu

JavaScript je jazyk interpretovaný, je tedy překládaný za běhu a vykonáván podle svého zdrojového kódu. Syntaxe je tedy C-like a jazyk je dynamicky typovaný, obsahuje pouze jeden numerický typ a to number, typ string pro text, boolean pro pravdivostní hodnotu a object pro cokoliv jiného. Jazyk je objektově orientovaný, ale je zde velká zvláštnost v návrhu. Objekt je totiž to samé jako slovník, tedy obecný box. JavaScript využívá tzv. funkcionální paradigma, které umožňuje do běžné proměnné ukládat funkci. Tato zdánlivě jednoduchá vlastnost potom umožňuje předávat funkce v parametru jiné funkce (tzv. callback) nebo dokonce využít funkci jako konstruktor objektu (Obr. 9). Objektově orientované programování zde nabývá úplně nových rozměrů, funguje zde prototypová dědičnost, tedy objekt je předlohou jiného objektu [7].

```
// var tolerance = cachedParameters["operation:tolerance"];
 var stockToLeave = cachedParameters["operation:stockToLeave"];
 var axialStockToLeave = cachedParameters["operation:verticalStockToLeave"];
 var maximumStepdown = cachedParameters["operation:maximumStepdown"];
 var maximumStepover = cachedParameters["operation:maximumStepover"] ? cachedParameters
 var optimalLoad = cachedParameters["operation:optimalLoad"];
 var loadDeviation = cachedParameters["operation:loadDeviation"];
```

Obr. 9: Předání funkce pomocí parametru.

V prostředí postprocesoringu je JavaScript rozdělen do několika sekcí, z nichž každá splňuje svoji funkci v určité části seřizovacího listu, nebo v jeho vizuálním nastavení. Každá z těchto sekcí je pojmenována podle svojí funkce, která je definována ve výpočtovém jádru CAM systému, ze kterého dostáváme informace pomocí skupin a CL dat.

3.2 Co je to API

Zkratka API je používána v softwarovém inženýrství a rozumí se tím rozhraní pro aplikace a jejich programování. Jedná se o balík knihoven, funkcí, procedur, protokolů a tříd, které mohou programátoři používat pro komunikaci se softwarem. Funkce rozhraní jsou na bázi programových celků, které programátoři používají a nemusejí je tak sami programovat. Cílem API je komunikace mezi 2 aplikacemi, kterým je tak umožněna výměna dat [9]. Komunikaci lze nastavit jak jednosměrnou, tak i obousměrnou.

API rozhraní existuje na 3 úrovních:

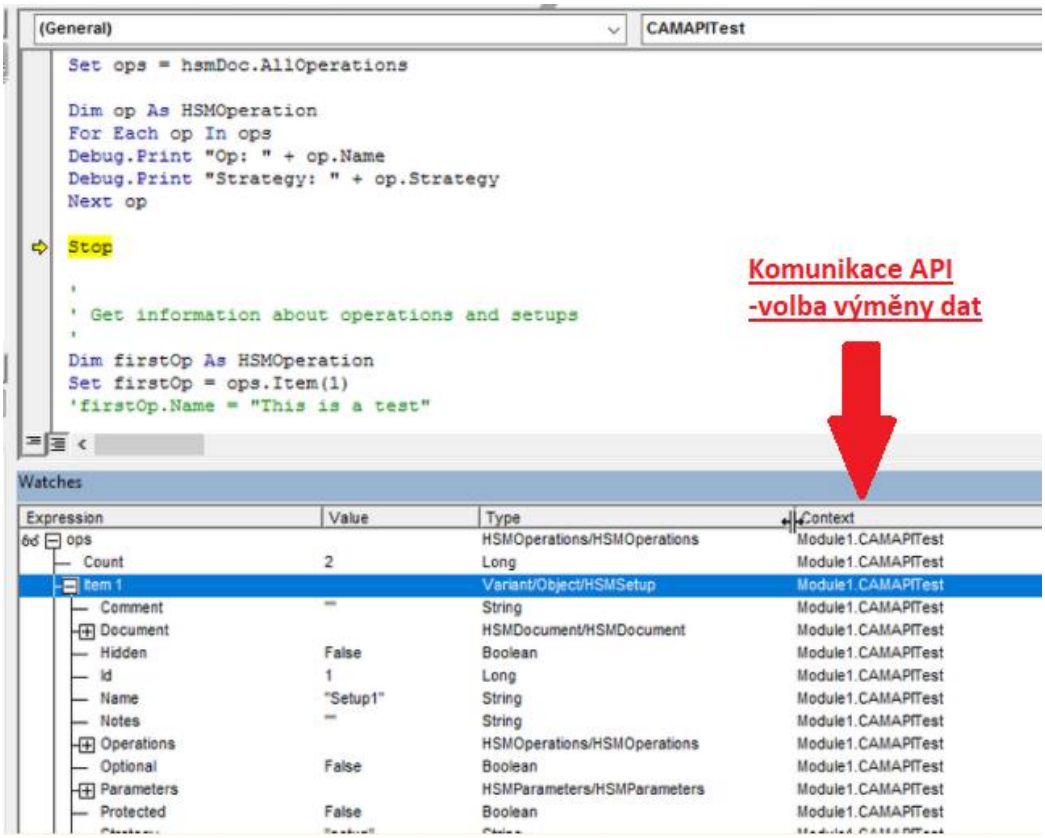
- operační systém,
- konkrétní programy,
- webové služby.

Nejvíce využívána jsou API grafická, kam se řadí DirectX nebo OpenGL. Dále existují API operačních systémů, kde mezi nejrozšířenější patří POSIX a Win32. Pro operační systémy Microsoft Windows je používáno jednotné Windows API, které nerozlišuje systémové volání a volání knihovních funkcí.

API lze dále rozdělit na:

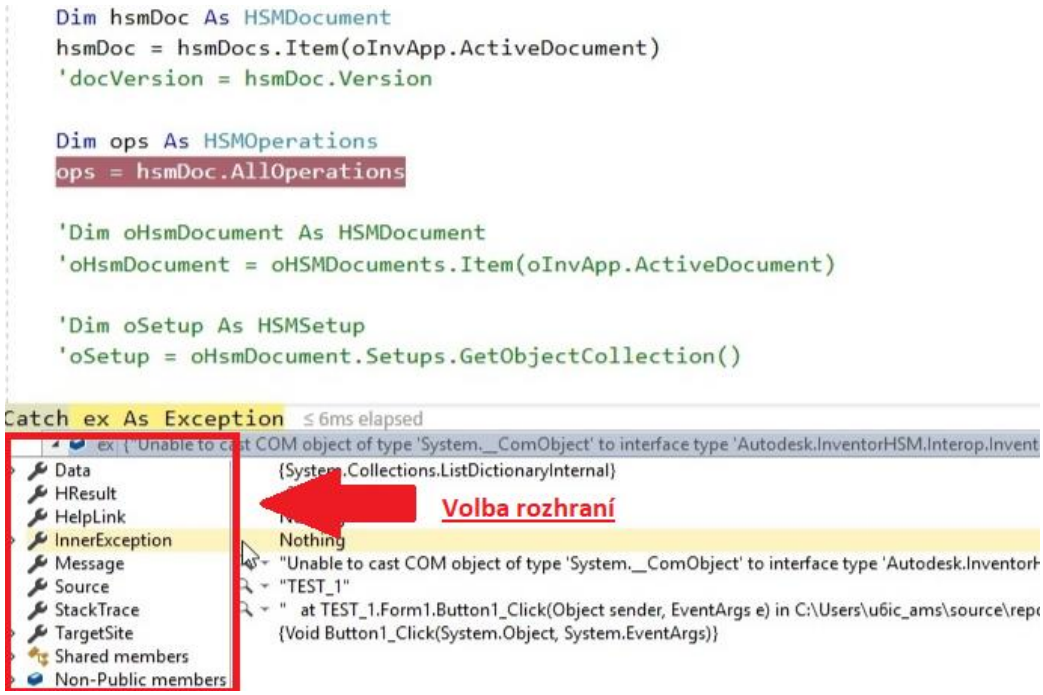
- Obecné – úplný soubor rozhraní API (Java API),
- Konkrétní – řeší konkrétní roblémy (Google Maps),
- Jazykově závislé – jeden programovací jazyk,
- Jazykově nezávislé – více programovacích jazyků.

V praxi to vypadá tak, že uživatel obsluhující daný CAD SW, má možnost ho rozšířit o jisté funkce. A právě pro takové rozšíření je využíváno API, se kterým je to snadné. Vždy záleží přímo na tom, co kdo chce z externí aplikace získávat a naopak, co kdo chce do aplikace zasílat. Komunikace pak probíhá formou vzdálené aplikace pomocí speciálních formátů, jejíž úkolem je výměna dat (Obr. 10).



Obr. 10: API komunikace.

API může být používán k odkazování na kompletní rozhraní, jednu funkci nebo dokonce na soubor více API poskytovaných SW. Proto je rozsah výzkumu obvykle určen osobou nebo dokumentem, který sděluje informace (Obr. **11**) [6].



Obr. 11: Volba rozhraní.

3.2.1 Implementace

Standard POSIX definuje API, který pokrývá širokou škálu běžně používaných funkcí a umožňuje je používat na mnoha různých systémech (typicky unixové systémy – Linux, Mac OS X, různé BSD systémy atd.). Použití tohoto API však vyžaduje na každé platformě danou aplikaci znovu přeložit (překompilovat). Kompatibilní API však umožňuje naprogramovat jeden zdrojový kód tak, aby beze změn fungoval na libovolném systému implementujícím toto API. Přenositelnost je výhodná jak pro tvůrce software (již existující produkt lze snadno přenést na jinou platformu), tak pro uživatele (mohou instalovat starší software na své novější systémy bez nutnosti zakoupení upgrade), i když to často vyžaduje přenášet i doplňující knihovny [5].

Microsoft pro vytvoření zpětné kompatibility Windows API (Win32, tj. pro možnost běhu starších aplikací na novějších verzích systému Windows) poskytuje možnost nastavit specifický „režim kompatibility“. Apple zastává v tomto ohledu méně vstřícný postoj, což mu poskytuje větší svobodu při vývoji aplikací za cenu označení starších programů jako „zastaralých“.

Mezi unixovými operačními systémy existuje mnoho příbuzných, ale navzájem nekompatibilních operačních systémů, přestože běží na společné technické platformě (zejména IBM PC kompatibilní systémy). Bylo zde mnoho pokusů o standardizaci API tak, aby výrobci software mohli šířit jedinou binární formu aplikace pro všechny tyto systémy, avšak do dnešního dne se žádný z nich nesetkal s větším úspěchem. Existuje projekt LSB (Linux Standard Base) pro platformu Linux a BSD (FreeBSD, NetBSD, OpenBSD), který by mohl zaručit různé úrovně kompatibility API a zajistit zpětnou kompatibilitu (pro programy napsané pro starší verze, které běží na novějších distribucích systému) a multiplatformní kompatibilitu (umožňuje spuštění kódu určeného pro jinou platformu bez nutnosti rekompilace).

Vzhledem k aktuálnímu rozsahu tohoto projektu nebude využito komunikačního rozhranní API. Jednotlivé scripty budou spojeny přes meziúložiště dat. Komunikační cesta těchto dat bude zohledněna v obou scriptech a popsána v praktické části projektu. Je předpokládán další vývoj tohoto projektu, a proto daná implementace komunikace přes úložiště může být v budoucnu integrována do prostředí API.

3.2.2 API v objektově-orientovaném jazyce

V objektově orientovaných jazycích API obvykle obsahuje popis souboru a definic tříd, dále pak chování spojených s těmito třídami. Chování je soubor pravidel, jak bude objekt (odvozený z této třídy) jednat v daných situacích. Tento abstraktní pojem je spojen se skutečnou exponenciální funkcí nebo jsou zpřístupněny pro třídy, které jsou prováděny, pokud jde o třídy metod.

API v tomto případě může být chápána jako souhrn všech způsobů veřejných tříd (obvykle nazvaná jako třída rozhraní). To znamená, že API stanoví metody, které zvládají objekty odvozené z definic tříd. Obecněji, jeden může vidět API jako sbírku všech možných objektů, které lze odvodit z definic tříd a jejich chování. Použití je opět zprostředkováno metodami, ale v tomto výkladu, jsou metody chápány jako technické detaily, jak je implementováno chování.

Například: třída představující zásobník může vystavit veřejně dvě metody push() (přidat nové položky do zásobníku), a pop()(extrahovat poslední položku, v ideálním případě umístěnou na vrcholu zásobníku) [10].

API v tomto případě mohou být vykládány jako dvě metody pop() a push() nebo obecněji jako představa, že lze používat položky typu zásobníku, který implementuje chování zásobníku (hromada vystavila své top přidat / odebrat prvky).

Tento koncept lze provádět na místě, kde třída rozhraní API nemá metody pro všechno, ale pouze se s nimi asociuje. Například jazyk Java API obsahuje rozhraní Serializable, což je rozhraní, které vyžaduje třída, která implementuje její chování v serializovaném módě. To nevyžaduje mít žádné veřejné metody, ale vyžaduje, aby třída měla povolení zastoupení, které lze uložit kdykoliv. To obvykle platí pro všechny třídy containing, které jsou jednoduché a datové a nemají žádnou vazbu na externí zdroje nebo jedno otevřené připojení k souboru, vzdálený systém nebo externí zařízení [10].

V tomto smyslu, v objektově orientovaných jazycích, API definuje sadu chování, případně zprostředkovaný soubor třídy metod. V těchto jazycích je API stále distribuována jako knihovna. Například jazyk Java knihoven, zahrnující soubor API, který je poskytovány ve formě JDK používaný vývojáři k vytváření nových Java programů. JDK obsahuje dokumentaci API v notaci Javadoc. Kvalita dokumentace k API je často určujícím faktorem jeho úspěchu, pokud jde o snadnost použití.

3.3 Prostředí API Inventoru HSM

Je-li požadován přístup na kmenovou úroveň API Inventor HSM Addin (objekt InventorHSMAPI), musí být nejprve vyhledán objekt AddM HSM Inventor ve vlastnosti ApplicationAddins aplikace Inventor. S tímto je získán přístup k jeho vlastnosti Automation, která bude instancí InventorHSMAPI.

Přístup k API:

- získání kolekce HSMDocuments,
- přístup k HSMDocumentu přidruženému k aktivnímu dokumentu aplikace Inventor,
- získání nastavení a operací v HSMDocumentu,
- generování dráhy nástroje pro celý dokument nebo konkrétní operace,
- následné zpracování.

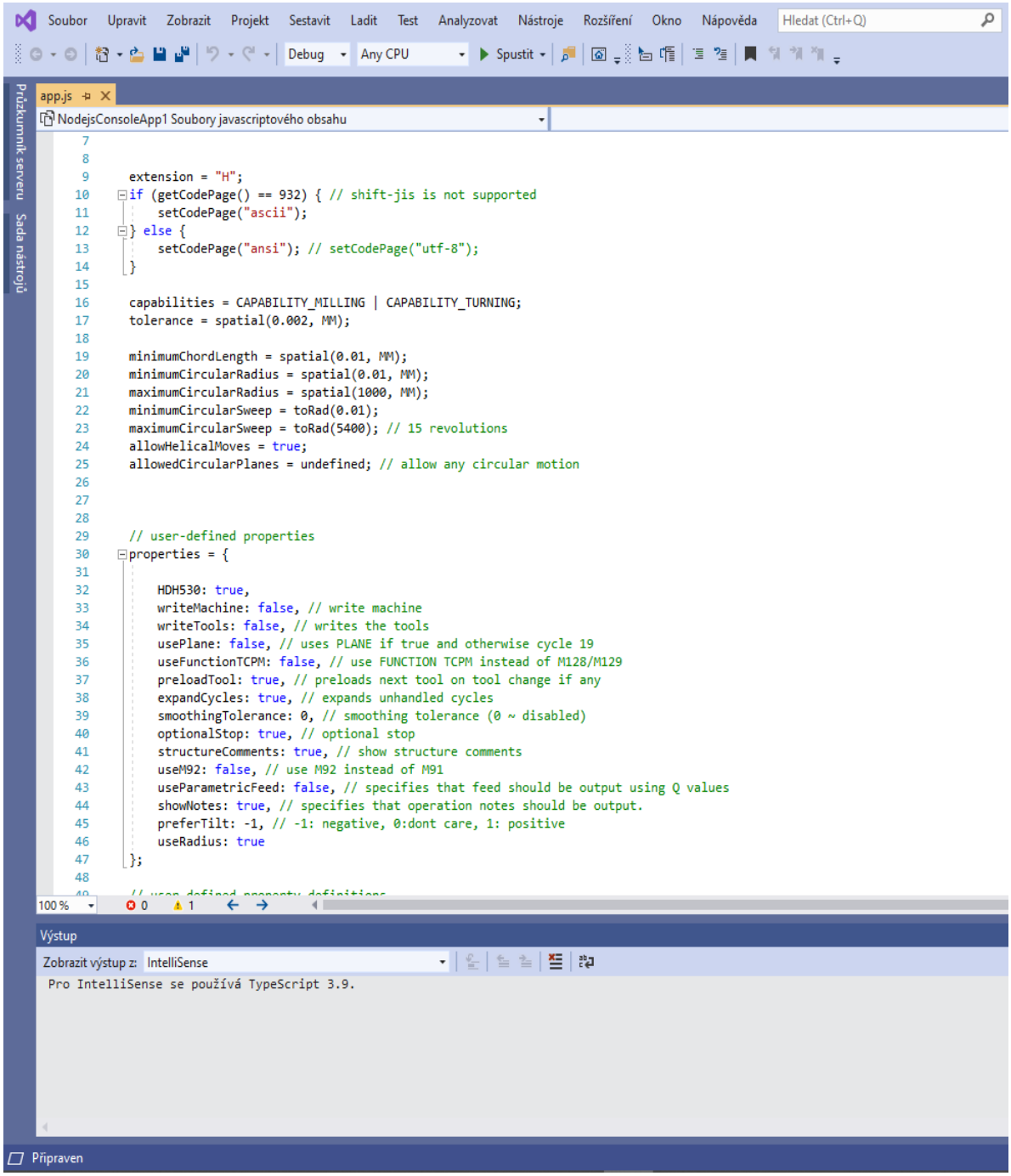
Objekt InventorHSMAPI představuje nejvyšší úroveň API. Je k němu přistupováno prostřednictvím vlastnosti Automation doplňkového objektu Inventor HSM.

Z tohoto objektu může být přistupováno k různým informacím o doplňku a také k jakýmkoli dostupným objektům HSMDocument, které poskytují přístup k datům CAM uchovávaným v otevřených dokumentech aplikace Inventor.

3.4 Rozhraní InventorHSM

Program Autodesk HSM umožňuje využívat 64bitovou architekturu a „hyper-threading“ (vícejádrové a víceprocesorové stanice, až 24 jader), což se jednoznačně projeví snížením času potřebného pro zpracování náročných úloh. Autodesk HSM nabízí automatické využití distribuovaného CAM serveru pro rychlejší výpočty v síti LAN (nevyužité počítače v síti). Do toho spadá i manažer operací, kde lze jednoduše navolit, kdy a jaké operace mají být zpracovány, a to třeba i na pozadí, čímž je možné se věnovat navazující technologii.

4 Vlastní programování aplikace



Obr. 12: Prostředí Visual Studio.

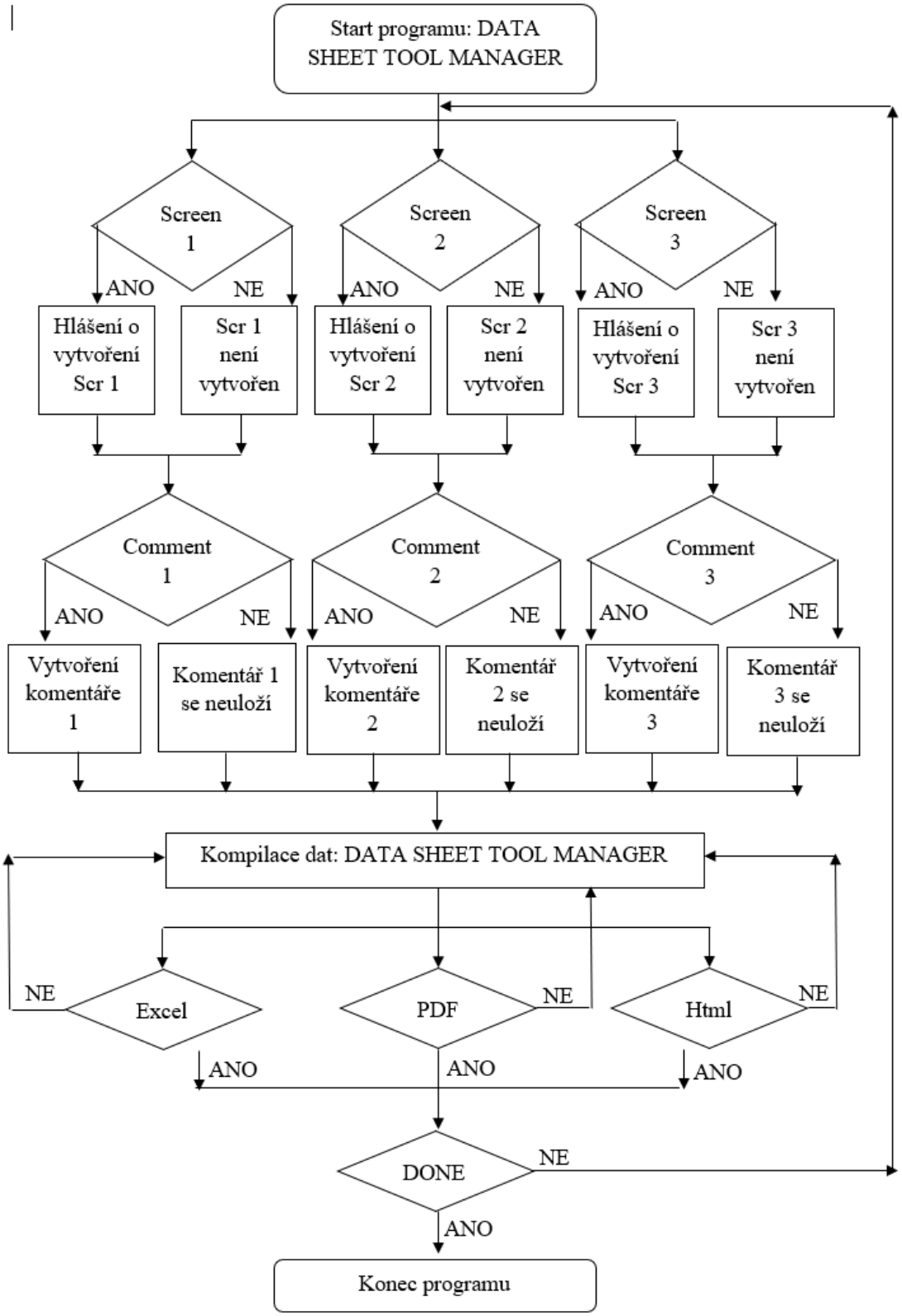
Aplikace, která bude spouštět generování seřizovacího listu a doplňovat jej o přidané obrázky a komentáře bude tvořena ve vývojovém prostředí C#. Tento jazyk byl volen vzhledem k aktuálnímu rozsahu projektu. Pro další vývoj projektu by bylo vhodné zvolit jiný programovací jazyk, jako třeba JavaAPI, který je kompatibilnější s daným prostředím

pro případné nadstavby. Vše bude vytvářeno ve Visual Studio (Obr. 12). Visual Studio je populární vývojové prostředí Microsoftu, které je standardem pro programování nových aplikací pro Windows. Dnes velký komerční balík doplňuje také bezplatná verze Community a multiplatformní editor s doplňky Visual Studio Code.

Programovací jazyk C# byl zvolen z důvodu aktuální velikosti tohoto projektu. Je zde opuštěna myšlenka původního návrhu vytvořit kompletně přidanou aplikaci v prostředí API. Jelikož se bude jednat o funkce vytvoření SCREENŮ a editaci POZNÁMEK, není nutné zde navazovat komunikaci s prostředím HSMapi. Pro tyto funkce bude postačující programovací jazyk C# a funkce prostředí MS. Funkce a příkazy prostředí HSMapi jsou často využívány při práci s parametry a informacemi z prostředí InventorHSM, tedy i její modelovací částí a práci s 3D objekty. Samozřejmostí je i návaznost na prostředí CAM, kde lze pracovat například s operacemi a strategiemi.

Aplikace bude vytvořena samostatně (C#) a následně navázána na script daného seřizovacího listu (JavaScript). Aplikace jako taková včetně jejích zdrojových souborů a příloh, bude uložena v samostatné složce instalačních souborů InventorHSM.

4.1 Vývojový diagram



Obr. 13: Vývojový diagram

4.2 Postup řešení jednotlivých sekcí projektu

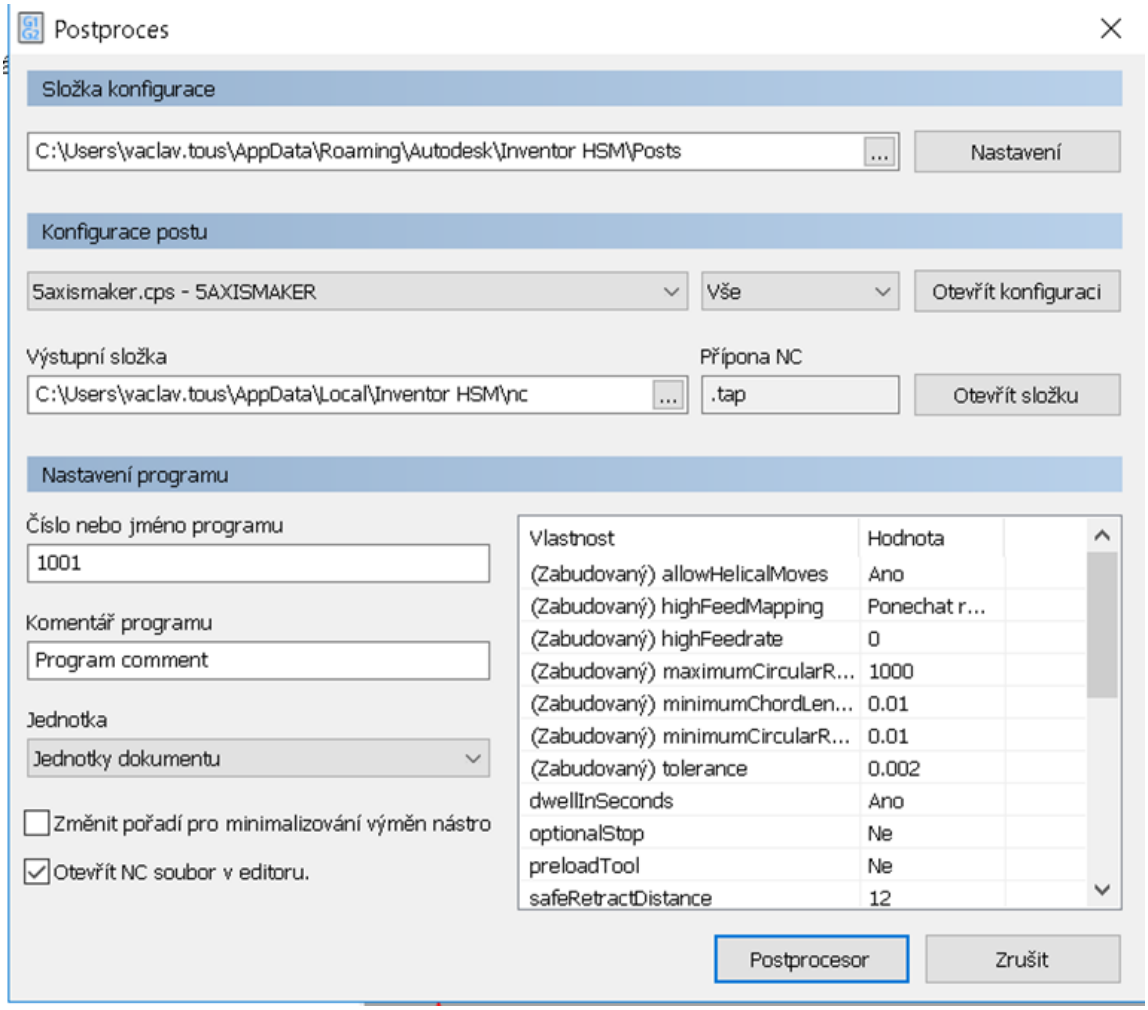
V první části projektu byl upraven script seřizovacího listu, který je psán v jazyce JS. Tato část byla volena jako první hned z několika důvodů. První z nich je pracnost a časová náročnost v daném projektu. Z celého projektu tato část tvoří cca 70 %. Ačkoliv se o této práci bavíme jako o vytvoření aplikace pro generování seřizovacího listu (a ne úpravu scriptu seřizovacího listu) je nutné zde zmínit, že script seřizovacího listu je ve své podstatě jádrem celé aplikace. Jinak řečeno, jádro aplikace je tvořeno ze 70 % scriptem seřizovacího listu.

Další částí bude tvorba aplikace jako takové. Vytvoření návrhu aplikace a jejího účelu, definice proměnných, tvorba vizuální části aplikace, popřípadě doladění výstupního designu. Vytvořená aplikace bude mít funkci nejen jako spouštěče automatického generování seřizovacího listu, ale bude jej doplňovat o případné screeny, které jsou možné vytvořit během simulace obrábění. Tyto screeny budou zařazeni na další stránku seřizovacího listu i s komentáři, které je možné zanést již v prostředí aplikace (Obr.17).

V závěru tohoto projektu bude nutné vytvořit komunikaci mezi těmito dvěma jazyky, jelikož komunikace mezi dvěma rozdílnými jazyky napřímo není možná. Jak již bylo zmíněno, bylo upuštěno od prostředí HSMapi, jelikož v tomto případě to není nutné. Bude zde definováno meziúložiště pro obě tyto aplikace, které bude fungovat jako úložný box pro zdrojová data. Vytvořená aplikace bude tento box naplňovat vytvořenými screeny a poznámkami, tedy jednoduchým textem a postprocesor seřizovacího listu bude tento box využívat jako zdrojovou složku potřebných dat pro výstup do seřizovacího listu.

4.3 Úprava scriptu seřizovacího listu

Script seřizovacího listu je psán v jazyce JavaScript. Jedná se o stejný druh souboru jako pro postprocesoring NC kódu. Na rozdíl od běžného postprocesoringu, kde jsou řešeny především přípravné funkce a dráhy nástroje, je zde více variabilních prvků, které lze upravovat, popřípadě definovat jiným způsobem. NC postprocesoring je omezen řídicím systémem a kód musí splňovat veškeré jeho atributy, aby byl funkční. DATA SHEET postprocesoring je omezen funkcemi JavaScriptu.



Obr. 14: Okno pro vkládání postprocesorů

Tlačítko Postproces po rozkliknutí otevře okno (Obr. 14), ve kterém je možné vkládat data postprocesorů v příslušném formátu. Tyto postprocesory mohou být jak pro NC postprocesoring, tak pro DATA SHEET postprocesoring. Obsahuje knihovnu jak se základními postprocesory od společnosti Autodesk, tak i s osobními postprocesory, definování úložiště dat postprocesorů, jejich názvů a komentářů [14]. V tomto okně

je možné otevřít konfiguraci a provádět úpravy postprocesoru (Příloha 1). Verze 2018 již obsahuje i online knihovnu, která je veřejně dostupná na stránkách Autodesku.

4.3.1 DATA SHEET-programování a úprava scriptu

4.3.1.1 Definice nových výstupů

Jak již bylo zmíněno, finální výstup bude mít pouze dvě sekce z původních tří. Samostatná sekce NÁSTROJE bude odstraněna a informace z ní budou integrovány do jednotné sekce OPERACE. Z výsledných dvou sekcí budou odstraněny, respektive nahrazeny následující parametry viz Tab. 1.

Sekce	**Původní verze**	**Nová verze**
	Výstup: *společné parametry*, *rozdílné parametry*	
Hlavička	Díl Spodek polotovaru od počátku Vršek polotovaru od počátku Délka obrobení Délka rychloposuvu Počet operací Počet nástrojů Nástroje Odhadovaný čas cyklu Název	Polotovar Počet operací Počet nástrojů Nástroje Maximální posuv Odhadovaný čas cyklu Název + obrázek
Operace	Popis Strategie Počátek Chladící médium Čas operace Korekce T, D, L Typ nástroje Průměr nástroje Délka nástroje Počet břitů	Popis Strategie Počátek Chladící médium Čas operace Korekce T, D, L Typ nástroje Průměr nástroje Délka nástroje Počet břitů

	Držák nástroje Poznámka Tolerance Stranový krok Krok dolu Délka obrábění Délka rychloposuvu Maximální otáčky Maximální posuv	Držák nástroje Poznámka **+ Volné kapacity** **pro vlastní návrhy**

Tab. 1: Původní verze a nová verze.

4.3.1.2 Úprava/nahrazení/odstranění výpisů DS

```
var maximumFeed = 0;
var maximumSpindleSpeed = 0;
var cuttingDistance = 0;
var rapidDistance = 0;
var cycleTime = 0;
for (var j = 0; j < getNumberOfSections(); ++j) {
  var section = getSection(j);
  if (section.getTool().number == tool.number) {
    maximumFeed = Math.max(maximumFeed, section.getMaximumFeedrate());
    maximumSpindleSpeed = Math.max(maximumSpindleSpeed, section.getMaximumSpindleSpeed());
    cuttingDistance += section.getCuttingDistance();
    rapidDistance += section.getRapidDistance();
    cycleTime += section.getCycleTime();
  }
}
```

Obr. 15: Deklarace proměnných – ukázka zápisu.

Na Obr. 15 je část scriptu, kde máme deklarovány původní proměnné pomocí VAR. Jejich výpis do výstupního formátu listu lze omezit jednoduchým zakomentováním těchto proměnných viz Obr. 16, popřípadě jejich úplným odstraněním. Jelikož kód funguje jako celek, jejich deklarace na začátku platí pro celý script. Je nutné dobře rozmyslet, jakým způsobem chceme tento výstup eliminovat. Musí být počítáno s tím, že při případných dalších úpravách musí zůstat script přehledný, pokud možno úplný. Tím je myšleno, že není nutnost odstraňovat celé části kódu, lze je pouze zakázat.

Na Obr. 16 je omezena funkce DOCUMENT-PATH automatického generování názvu součásti do hlavičky seřizovacího listu. Ve scriptu je psána podmínka: pokud bude

existovat parametr NÁZEV SOUČÁSTI, bude tento název generován do seřizovacího listu. Zakomentováním této podmínky byl zakázán výstup NÁZVU SOUČÁSTI do seřizovacího listu. Na místě, kde byl tento parametr generován bude prázdná kolonka. Pokud bychom chtěli zakázat celkovou definici názvu součásti, je nutné ve scriptu JS zakomentovat celou funkci (Obr. 16). V tomto případě se nám v seřizovacím listu neobjeví ani popis, co mělo být generováno. Stejným způsobem lze zakázat další funkce, například výpis maximálních a minimálních velikostí polotovaru (Obr. 17).

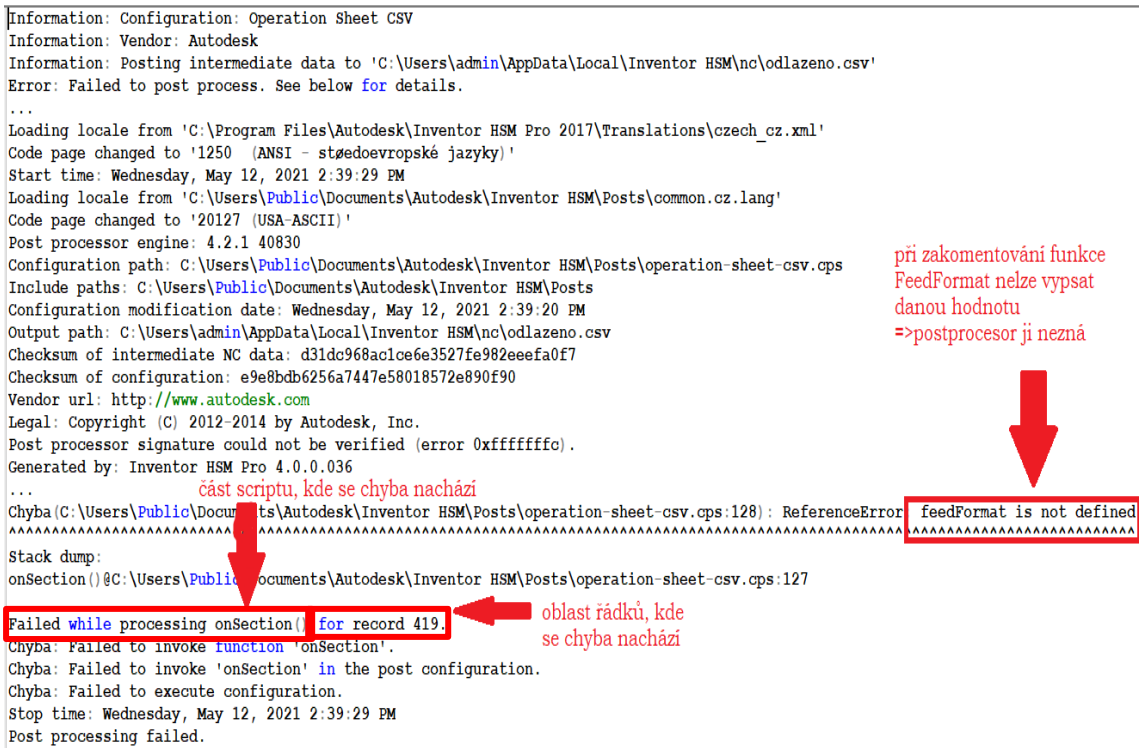
```
ZAKOMENTOVANI CELE FUNKCE
/*
    if (hasParameter("iso9000/document-control")) {
      var id = getParameter("iso9000/document-control");
      if (id) {
        c += makeRow(makeColumn(d(localize("Job ISO-9000 Control") +
      }
    }
*/

ZAKOMENTOVANI PODMINKY
//    if (properties.showDocumentPath) {
//      if (hasParameter("document-path")) {
        var path = getParameter("document-path");
//        if (path) {
          c += makeRow(makeColumn(d(localize("Document Path") + ": ")
        }
      }
```

Obr. 16: Zakomentování funkcí/podmínek

```
/*
  if (is3D()) {
    var zRange = currentSection.getGlobalZRange();
    c2 += makeRow(makeColumn(d(localize("Maximum Z") + ": ") + v(spatialFormat.format(zRange.getMaximum()) + getUnitSymbolAsString())));
    c2 += makeRow(makeColumn(d(localize("Minimum Z") + ": ") + v(spatialFormat.format(zRange.getMinimum()) + getUnitSymbolAsString())));
  }
*/
```

Obr. 17: Odstranění MAX Z/MIN Z z původní hlavičky seřizovacího listu

I při tomto procesu je třeba brát v potaz, že proměnné, které jsou deklarovány v hlavičce scriptu, se mohou objevit ve kterékoliv části kódu. A proto je nutné projít celý script a zakomentovat i veškeré další výpisy těchto proměnných (Příloha 1). Pokud by došlo k přehlédnutí těchto proměnných, není možné vygenerovat výsledný formát seřizovacího listu. **Dojde k chybovému hlášení** (Obr. 18).



Obr. 18: Chybové hlášení.

Tímto způsobem je nutné postupovat celou dobu, abychom dosáhli požadované funkčnosti. Platí to nejen pro definici, odstranění, zakomentování nebo jinou práci s proměnnými, ale i pro ostatní úpravy, včetně vizuální stránky výstupního listu.

4.3.1.3 Deklarace nových proměnných

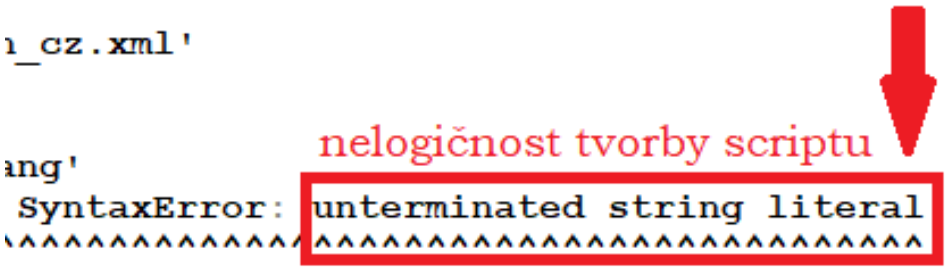
Při definici nových proměnných je nutné dávat velký pozor při jejich deklaraci. U názvů musíme počítat s ostatními proměnnými, které již daný script obsahuje. Kdyby došlo k jejich shodě, mohlo by nastat více situací, z nichž ani jedna není žádoucí (Obr. 19).

Příklad:

Script nebude funkční z důvodu nelogičnosti zápisu. Pro dvě různé funkce bude deklarována jedna proměnná. Tato proměnná je vypsána v různých částech scriptu s různými podmínkami, vlastnostmi a účely. Pracujeme tedy s jednou proměnnou na dvou odlišných úkolech, které spolu nijak nesouvisí, nebo si přímo odporují => CHYBOVÉ HLÁŠENÍ.

Příklad:

Script vezme v potaz obě proměnné, jelikož si navzájem neodporují. Oproti předchozímu případu může nastat, že dvě různé proměnné, avšak se stejným názvem nemusí být zapodmínkovány, nebo určitým způsobem modifikovány. Mohou mít funkci jednoduchého výpisu zvoleného parametru. Script je tedy funkční, dostaneme finální výstup, avšak obsah není správný => CHYBNÝ VÝSTUP.



Obr. 19: Chyba logiky scriptu.

Název proměnné musí začínat písmenem, dolarem ‚$‘, nebo podtržítkem ‚_‘. Nesmí začínat číslicí. Toto omezení počítačům usnadňuje rozlišování čísel a identifikátorů (tj. názvů proměnných, funkcí apod.). Co se týče písmen, je dovoleno používat české znaky (písmena s diakritikou), doporučuje se však, a je to i běžnou praxí, používat výhradně písmena anglické abecedy. I přes to, že to komplikuje psaní českých názvů proměnných.

Ve zbytku názvu proměnné už mohou být i číslice. Rozhodně tam však nepatří spojovník ‚-‘ ani žádné jiné znaky, které jsou v JavaScriptu interpretovány jako operátory. A samozřejmě názvem proměnné nesmí být žádné ze slov, která jsou součástí syntaxe JavaScriptu, tedy klíčová slova.

Délka názvu proměnné je libovolná, je tedy možné ji pojmenovat pouze jedním písmenem. Ve většině případů je však lepší volit delší názvy, které dobře vystihují

význam dané proměnné, klidně i víceslovné. V názvu proměnné sice nejsou povoleny mezery, ale lze si pomoci jinak, například slova začínat velkými písmeny nebo jako oddělovač použít podtržítko.

V JavaScriptu existují tři druhy proměnných (plus ještě konstanty), které se liší oborem platnosti:

- globální proměnné,
- lokální proměnné,
- proměnné s působností v rozsahu bloku (*block-scoped*).

Pokud je proměnná deklarována mimo funkci, jedná se o globální proměnnou, která platí v celém skriptu. Je-li deklarována uvnitř funkce, jedná se o lokální proměnnou, k níž existuje přístup pouze uvnitř této funkce nebo uvnitř funkcí deklarovaných v této funkci (vnitřní funkce tzv. *closures* si totiž ukládají odkazy na lokální proměnné své mateřské funkce).

Globální i lokální proměnné mohou vznikat i bez explicitní deklarace. Nedeklarované proměnné se v JavaScriptu, není-li přepnut na striktní režim, automaticky interpretují jako globální. To je však spíš nevýhoda, protože případný překlep v názvu proměnné je brán jako nová proměnná, a tedy nezpůsobí ukončení skriptu a chybovou hlášku, takže skript běží dál – ovšem jinak, než jeho autor původně zamýšlel. Za lokální proměnné jsou zase považovány tzv. argumenty – pojmenované parametry funkce, které představují vstupní data.

Kromě těchto proměnných se do JavaScriptu později dostaly ještě proměnné a konstanty, jejichž obor platnosti je v kódu vymezen složenými závorkami „`{}`“. Deklarují se klíčovými slovy „let a const“. Hodnoty konstant pochopitelně nelze přepisovat – z toho také vyplývá, že hodnota musí být konstantě přidělena hned při její deklaraci.

```
const PERSON_NAME = "Václav Touš";
```

Konstanty byly do JavaScriptu zavedeny kvůli bezpečnosti, aby programátoři měli možnost definovat výchozí hodnoty, které by nemohly být přepsány cizím externím skriptem. Proto není dovoleno je znovu deklarovat. Také není povoleno deklarovat ve stejném oboru proměnnou nebo funkci, která by měla s konstantou společné jméno.

```
const x = true;
```

```
var x = false; //=>CHYBOVÉ HLÁŠENÍ
```

V tomto případě lze použít funkci localize, která nám odkazuje na hodnotu nebo výraz určité proměnné. Nemusíme ji vždy v dané části kódu znovu deklarovat, stačí se na ni pouze odkazovat viz (Obr. 20). Musíme také počítat s překladovým souborem do českého jazyka. Pokud chceme do seřizovacího listu přidat nějaké informace a chceme přitom použít svůj vlastní název, je nutné jej zaevidovat do překladového souboru. Tento soubor je ve formátu _.txt a obsahuje knihovnu názvů. Můžeme je rozdělit dle uplatňované sekce nebo vytvořit jeden globální. V tomto případě je použit lokální překladový soubor, který obsahuje knihovny pro prostředí HSM.

```
if (hasParameter("operation-strategy")) {
  var strategies = {
    drill: localize("Drilling"),
    face: localize("Facing"),
    path3d: localize("3D Path"),
    pocket2d: localize("Pocket 2D"),
    contour2d: localize("Contour 2D"),
    adaptive2d: localize("Adaptive 2D"),
    slot: localize("Slot"),
    circular: localize("Circular"),
    bore: localize("Bore"),
    thread: localize("Thread"),

    contour_new: localize("Contour"),
    contour: localize("Contour"),
    parallel_new: localize("Parallel"),
    parallel: localize("Parallel"),
    pocket_new: localize("Pocket"),
    pocket: localize("Pocket"),
    adaptive: localize("Adaptive"),
    horizontal_new: localize("Horizontal"),
    horizontal: localize("Horizontal"),
    flow: localize("Flow"),
    morph: localize("Morph"),
    pencil_new: localize("Pencil"),
    pencil: localize("Pencil"),
    project: localize("Project"),
    ramp: localize("Ramp"),
    radial_new: localize("Radial"),
    radial: localize("Radial"),
    scallop_new: localize("Scallop"),
    scallop: localize("Scallop"),
    morphed_spiral: localize("Morphed Spiral"),
    spiral_new: localize("Spiral"),
    spiral: localize("Spiral"),
    swarf5d: localize("Multi-Axis Swarf"),
    multiAxisContour: localize("Multi-Axis Contour")
  };
```

Obr. 20: Funkce LOCALIZE-v tomto případě na obráběcí strategie.

4.3.1.4 Eliminace sekcí

V původním seřizovacím listě byly sekce ve scriptu rozděleny jako samostatné proměnné. Tyto proměnné fungují jako funkce v postprocesoringu. Lze se na ně odkazovat, vkládat vnořené funkce, nebo nové proměnné. Každá tato funkce měla svůj název. V tomto případě c1, c2 a c3.

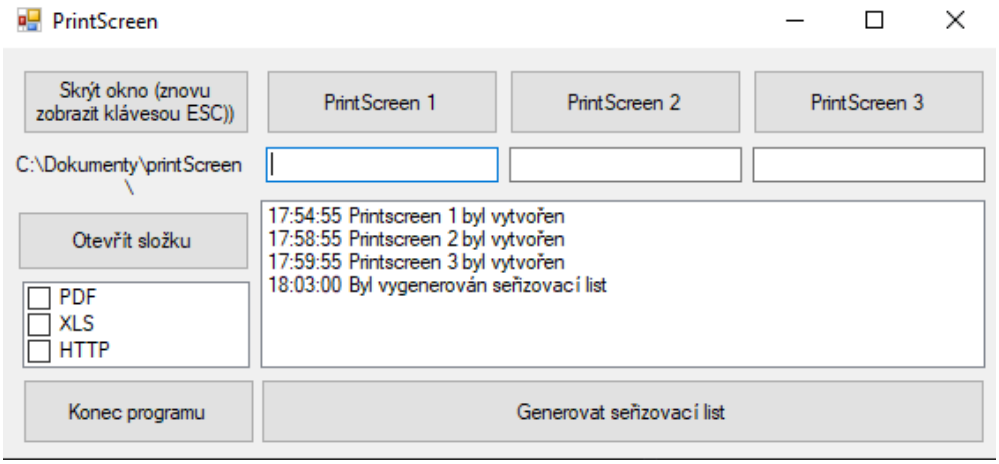
```
var c1 = "<table class=\"info\">";

var c2 = "<table class=\"tool\">";

var c3 = "<table class=\"info\" cellspacing=\"0\">";
```

Jelikož s nimi lze pracovat jako s funkcemi scriptu postprocesoringu, lze je také odstranit. Při jejich odstraňování musíme dodržovat zásady, které zde již byly definovány. Nesmíme zapomenout, že odkazy na tuto funkci budou neproveditelné a lze očekávat jedno z dalších chybových hlášení. Nejedná se pouze o odkazy přímo na funkci jako takovou, ale i na další funkce a proměnné, které byly deklarovány již v této vytvořené sekci (Obr. 21).

Obr. 21: Ukázka rozdělení na sekce v JS

4.4 Vytvoření aplikace pro generování seřizovacího listu¨

Námi požadovaná aplikace vytvořená v prostředí Visual Studia je vyobrazená na Obr. 22.



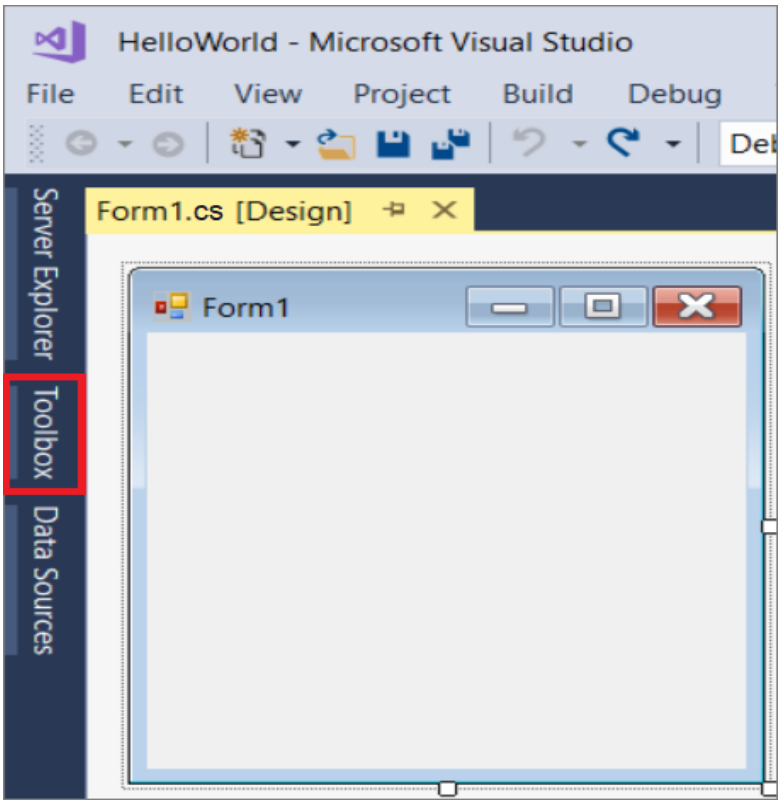
Obr. 22: Otevřená konzolová aplikace vytvořená v prostředí Visual Studia.

4.4.1 Programovací jazyk C#

C# neboli C sharp je vyvinutý firmou Microsoft. Jedná se o čistě objektový programovací jazyk, který slouží k vývoji aplikací pro prostředí .NET – dříve se jednalo pouze o nadstavbu operačního systému Windows. Nyní se .NET skládá ze čtyř komponentů: Visual Studio, jazyk, virtuální stroj (CLR) a knihovny. Získání tohoto jazyka je provedeno instalací IDE (Integrated Development Enviroment), který se nazývá Visual Studio 2019 Community Edition [17] [18].

4.4.2 Grafické uživatelské rozhraní (Windows Froms)

Grafické uživatelské rozhraní (GUI–Graphical User Interface) je v dnešní době neodmyslitelnou součástí téměř všech aplikací (Obr. 23). Jedná se o první framework z NETu, díky čemuž lze jednoduše tvořit aplikace pomocí grafického designeru. Základem tohoto rozhraní je většinou alespoň jedno okno nabízející uživateli určité informace. Při programování tohoto okna lze využít nespočet komponent, které na základě uživatelovy odezvy dokážou řídit běh celého programu. Protože prostředí .NET má své knihovny k vytváření Windows Forms, budeme využívat nástrojů Visual Studia [17] [18].



Obr. 23: Grafické uživatelské rozhraní VisualStudio pro tvorbu aplikace.

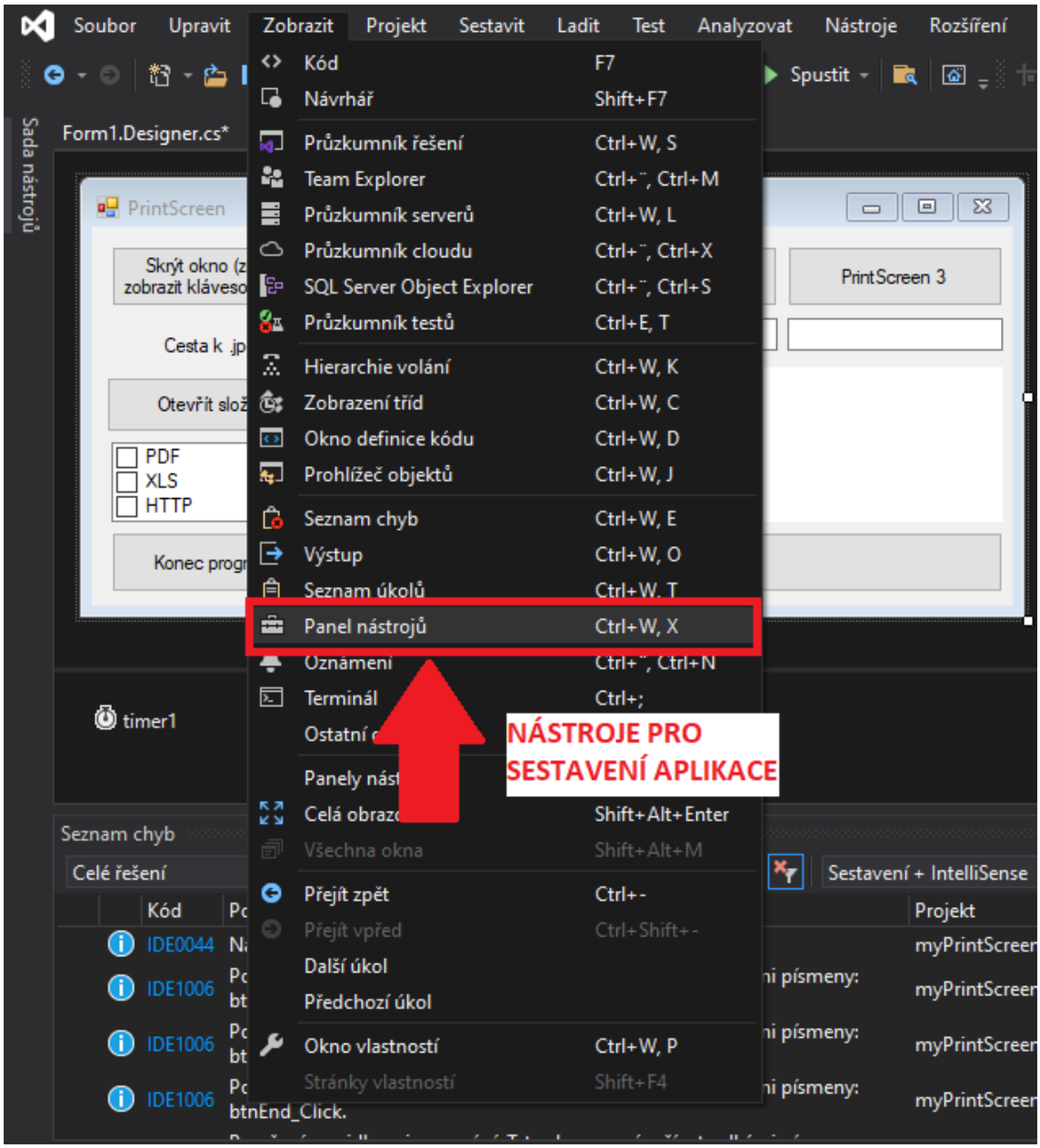
4.4.3 Projekt okenní aplikace

K vytvoření požadované aplikace postačí knihovna, definovaná ve jmenném prostoru System.Windows.Froms. Po otevření Visual Studia se založí nový projekt, který musí být patřičně pojmenován. Visual Studio 2019 vypíše seznam předdefinovaných projektů, volba Windows Forms App (.NET Core) vytvoří projekt [17].

Při vytvoření nového projektu se automaticky vygeneroval zdrojový kód, který jerozdělený do několika zdrojových souborů – Form1.cs a Program.cs, které najdeme v průzkumníku řešení. Form1.cs se skládá ze dvou souborů z nichž jeden se nazývá Form1.Designer. Visual Studio 2019 vygeneruje další soubory, ale ty pro tuto diplomovou práci nejsou klíčové, a proto je zde nebudu uvádět [18]. Zdrojový kód pro Form1.cs, Form1.Designer a Program.cs nalezneme v Příloha 2.

Mezi jeho nejdůležitější části pro úpravu slouží Designer, Properties a Toolbox. Designer (Grafický návrhář) slouží k vizualizaci formuláře. Properties (Vlastnosti) ukazují vlastnosti zvoleného prvku ve formuláři. Toolbox (Panel nástrojů) je seznam s jednotlivými ovládacími prvky [18].

Pro modifikování okenní aplikace lze použít Panel nástrojů (Obr. 24), ve kterém lze do okna přetáhnout buttony, labely, texboxy, aj. Při přetáhnutí do okenní aplikace se automaticky vygenerují vlastnosti nástrojů, které jsou obsaženy v souboru



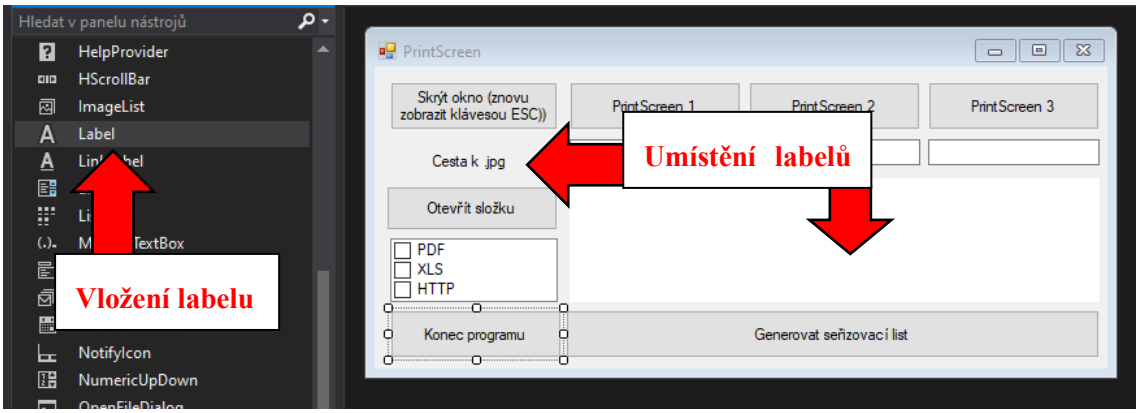
Form1.Designer.cs. Vlastnosti lze modifikovat v souboru Form1, kde se po pravé straně nacházejí Vlastnosti [17].

Obr. 24: Nástroje pro grafické sestavení aplikace.

4.4.3.1 Labely

Labely jsou textové popisky a jejich názvy by se měli měnit v případě, že je jich v aplikaci využíváno [18]. V programu jsou použity dvakrát (Obr. 25):

- Cesta k .jpg:
    - tento label nedělá nic jiného, než že vypíše textový popisek. Jeho text uživatel nemůže měnit.
- ProcessState:
    - slouží k zaznamenávání akcí ve vytvořené aplikaci, aby byl přehled o tom, zda tento program funguje správně. Přesněji řečeno tento label slouží k výpisu informací o stisknutí buttonů nebo zaškrtnutí v CheckedListBoxu.

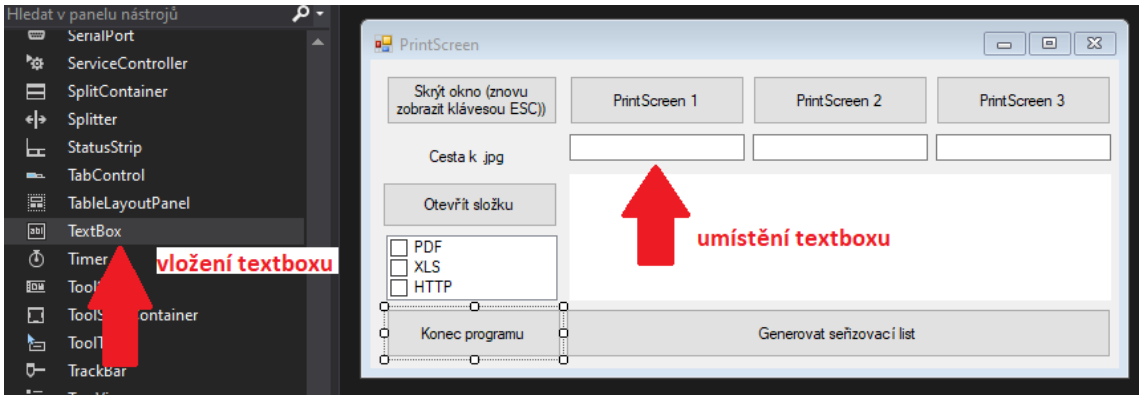


Obr. 25: Tvorba labelů ve Visual Studiu.

4.4.3.2 Textboxy

Textboxy slouží k zadávání textu uživatelem [18]. Tato aplikace obsahuje tři textboxy, které náleží buttonům PrintScreen 1-3. Do těchto textboxů může uživatel aplikace zapisovat poznámky a ty se dále vygenerují k požadovaným printscreenům. V programu jsou použity třikrát (Obr. 26):

- PrintScreen 1;
- PrintScreen 2;
- PrintScrren 3.



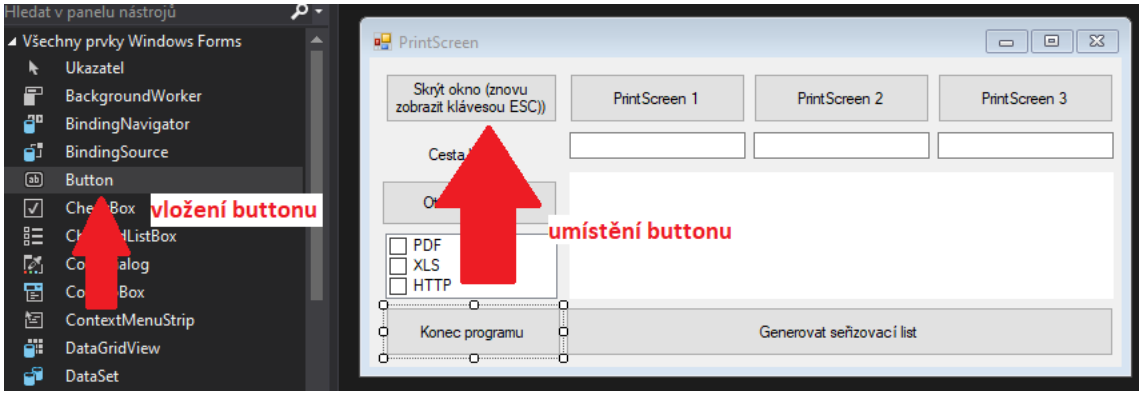
Obr. 26: Tvorba textboxů ve Visual Studiu.

4.4.3.3 Buttony

Buttony jsou určené k tomu, aby se po jejich stisknutí vyvolala akce (metoda) [18]. V této aplikaci se jedná o následující butony a jejich metody (Obr. 27):

- PrintScreen 1;
- PrintScreen 2;
- PrintScreen 3;
- Otevřít složku:
    - tento button nám otevře průzkumník a cestu k printscreenům, které jsme si vygenerovali;
- Generovat seřizovací list:
    - při zaškrtnutí formátů v CheckedListBooxu se nám vygeneruje seřizovací list. Pokud nebude nic vybráno, nic se nevygeneruje;
- Skrýt okno:
    - tento button nám umožní skrýt okno programu,
    - hodí se to v případě, kdy chceme vytvářet printscreeny,
    - aplikace tak běží na pozadí;
- Konec programu:
    - tímto buttonem se program ukončí,

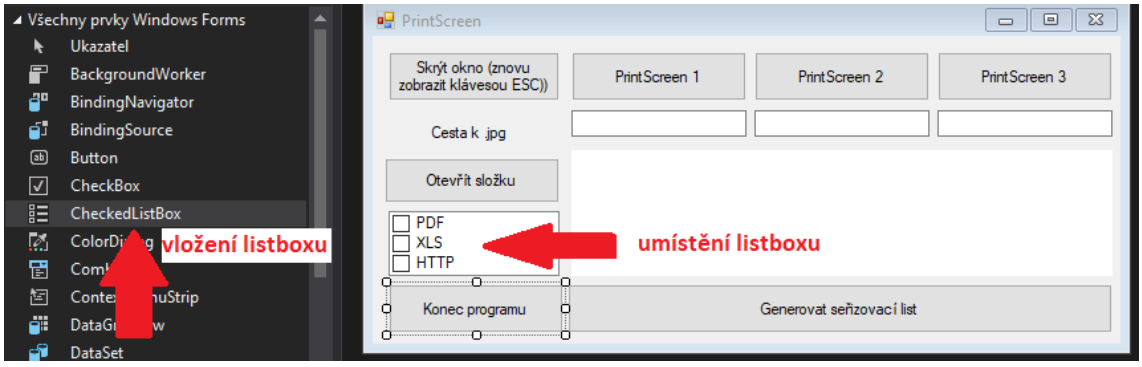
- v tomto případě můžeme program zavřít buttonem, který je vygenerovaný na začátku vytvoření projektu, a to červeným křížkem vpravo nahoře;



Obr. 27: Tvorba buttonů ve Visual Studiu.

4.4.3.4 CheckedListBox

Tento ovládací prvek slouží k označení daných možností o zaškrtnutí *true* nebo *false* u předem definovaného seznamu formátů (_.pdf, _.xls, _.http). V tomto programu byl použit pro zvolení formátu exportu z programu tento CheckedListBox z důvodu zachování přehlednosti a jednoduchosti volby formátu seřizovacího listu (Obr. 28) [18].



Obr. 28: Tvorba Listboxů ve Visual Studiu.

Grafická část je řešena v Desingeru, který obsahuje potřebné ovladače popsané výše. Tyto ovladače jsou v daném prostředí již nadefinovány a při jejich přidání do dané konzolové aplikace jsou automaticky generovány do vytvořeného scriptu (Obr. 29). Tyto proměnné lze upravovat, přidávat názvy a přemisťovat dle potřeby. Samotná práce s nimi je definována a omezena jejich obecnými vlastnostmi, ke kterým v dané aplikaci slouží.

Samotný kód tedy není kompletně psán ručně, ale následné navazující funkce a práce s těmito zadanými proměnnými ano.

```
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Button btnEnd;
        private System.Windows.Forms.Button btnHide;
        private System.Windows.Forms.Button btnFolder;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.CheckedListBox checkedListBox1;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.ListBox listBox1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.TextBox textBox3;
```

Obr. 29: Automaticky vygenerované proměnné ve Visual Studiu.

Na obrázku č.Obr. 30 lze vidět ruční psaní textu. V tomto případě se jedná o zápis funkce pro vytvoření bitmap a jejich evidence pomocí funkce bmpScreenshot.Save(saveFolder+namePicture). Nejen že je proveden grafický zápis, ale je i evidován pod příslušným pojmenováním. (namePicture). Další funkce řeší skrytí dané aplikace během práce v SW. Jak je vidět na obrázku č.Obr. 30, funkce this.ShowInTaskbar určuje, zda je daná aplikace zobrazena v prostředí Windows. V její základní fázi má hodnotu *false*, jelikož v počátku je nutné mít aplikaci zobrazenou pro její ovládání a editaci. Až po zadání příkazu SKRÝT APLIKACI nabudude hodnoty *true.* Po splnění této podmínky je aplikace skryta.

```
PixelFormat.Format32bppArgb);

            Graphics gfxScreenshot = Graphics.FromImage(bmpScreenshot);

            // Překopíruje obsah obrazovky do bitmapy
            gfxScreenshot.CopyFromScreen(Screen.PrimaryScreen.Bounds.X,

Screen.PrimaryScreen.Bounds.Y, 0, 0,

Screen.PrimaryScreen.Bounds.Size,

CopyPixelOperation.SourceCopy);

            // Uloží bitmapu do JPEG souboru
            bmpScreenshot.Save(saveFolder+namePicture);

        }

        /// <summary>
        /// skryje ovládací panel
        /// </summary>
        private void hideForm()
        {
             // Skryje aplikaci v pruhu úloh
            this.ShowInTaskbar = false;
            // Skryje okno aplikace
            this.Hide();
        }

        /// <summary>
        /// zobrazí ovládací panel
        /// </summary>
        private void showForm()
        {
            this.ShowInTaskbar = true;
            this.Show();
        }
       #endregion
    }
}
```

Obr. 30: Funkce uložení bitmap/Funkce skrytí aplikace na pozadí.

Daných screenů je zde možné definovat nekonečné množství, avšak v tomto případě jsou tři postačující. Pokud by měl být tento počet rozšířen, je nutné ho zohlednit již při základním grafickém tvoření aplikace a veškeré tyto scripty definovat.

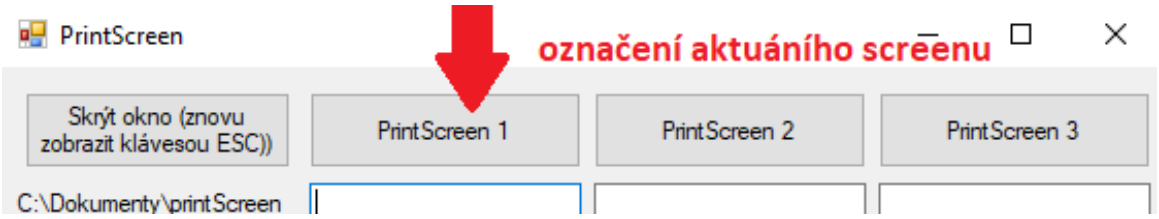
Pro opětovné zobrazení aplikace mezi těmito třemi screeny slouží uživateli klávesa ESC. V případě skrytí aplikace se grafické okno CAM SW nastaví dle potřeb vytvoření screenu. Stisknutím klávesy ESC dojde k vytvoření daného screenu a znovuotevření

konzolové aplikace. Tato funkce je volána pomocí CASE. Lze ji tedy vyvolat a zapodmínkovat v kódu, aniž bychom ji museli znovu definovat (Obr. 31).

```
// Snímá klávesy
foreach (System.Int32 i in Enum.GetValues(typeof(Keys)))
{
    if (GetAsyncKeyState(i) == -32767)
    {
        keyBuffer = Enum.GetName(typeof(Keys), i);

        // zpracování
        switch (keyBuffer)
        {
            case "PrintScreen":
                eventPrintScreen();
                return;

            case "Escape":
                showForm();
                return;
        }
    }
}
```

Obr. 31: Zdrojový kód funkce, která se zavolá po stiskutí klávesy escape.

Takto lze pokračovat i při tvorbě dalších screenů. V tomto případě je v aplikaci označen screen, který je aktuálně prováděn (Obr. 32). Pokud by v prostředí aplikace došlo k označení screenu, který byl již vytvořen, automaticky se přehraje nový obrázek a původní je nahrazen. Další funkcí je zde neúplnost zadání, tedy pokud není potřeba využít veškeré tři screeny. Aplikace bude fungovat i bez kompletního zadání. V seřozovacím listě tento screen nebude generován, zobrazí se pouze prázdné kolonky. Do budoucna je zde prostor pro grafické úpravy. Možný větší počet generovaných scriptů a jejich automatické generování bez zobrazení prázdných kolonek. Grafický výstup seřizovacího listu by byl automaticky upraven dle počtu generovaných obrázků.



Obr. 32: Aktuálně vytvářený screen ve Window Forms.

4.5 Problémy při tvorbě scriptů a jejich propojení

Jedním z častých problémů při tvorbě seřizovacích listů je jejich přehlednost. Skutečnost, že je psán v prostředí JS a každá sekce může být kdykoliv vyvolána, je zde velký prostor pro chybu špatně deklarovaných proměnných. Proměnné lze definovat pro celý script, nebo pro jednotlivé sekce. Při potřebě jejich výpisů tato kombinace může způsobit problémy při samotném generování postprocesoringu. Na Obr. 33 je špatně deklarována proměnná a její výpis způsobil celkovou kolizi scriptu (Obr. 34).

```
  var StockToLeave = cachedParameters["operation:stockToLeave"];
  var axialStockToLeave = cachedParameters["operation:verticalStockToLeave"];
  var maximumStepdown = cachedParameters["operation:maximumStepdown"];
  var maximumStepover = cachedParameters["operation:maximumStepover"] ? cachedParameters
  var optimalLoad = cachedParameters["operation:optimalLoad"];
  var loadDeviation = cachedParameters["operation:loadDeviation"];
/*
  if (t
    c1 = makeRow(makeColumn(d(localize("Tolerance") + ": ") + v(spatialFormat.format(to
  }
*/
  if (stockToLeave != undefined) {
    if ((axialStockToLeave != undefined) && (stockToLeave != axialStockToLeave)) {
```

rozdílně definované proměnné-chyba scriptu

Obr. 33: Chybně definované proměnné v prostředí JS.

```
Chyba (C:\Users\Public\Documents\Autodesk\Inventor HSM\Posts\setup-sheet.cps:692):
 ReferenceError: stockToLeave is not defined
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Stack dump:
onSectionEnd()@C:\Users\Public\Documents\Autodesk\Inventor HSM\Posts\setup-sheet.cps:691

Failed while processing onSectionEnd() for record 350.
Chyba: Failed to invoke function 'onSectionEnd'.
Chyba: Failed to invoke 'onSectionEnd' in the post configuration.
Chyba: Failed to execute configuration.
Stop time: Saturday, May 29, 2021 10:49:41 AM
Post processing failed.
```

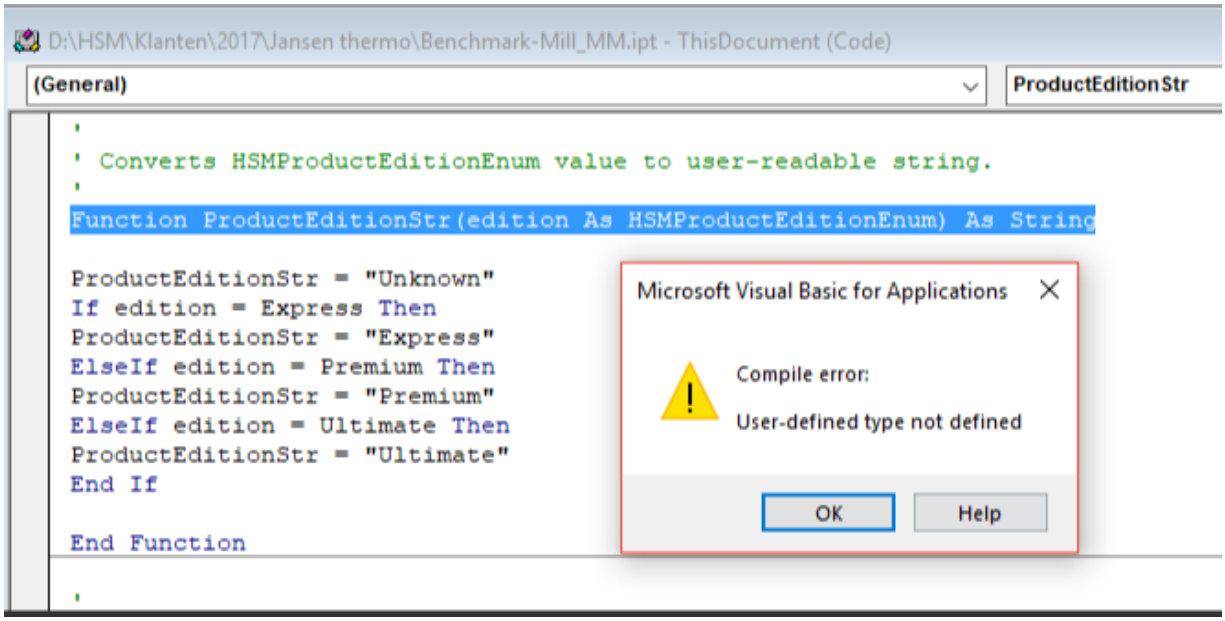
špatně deklarovaná proměnná

selhání celého scriptu

Obr. 34: Selhání scriptu po špatně definované proměnné v prostředí JS.

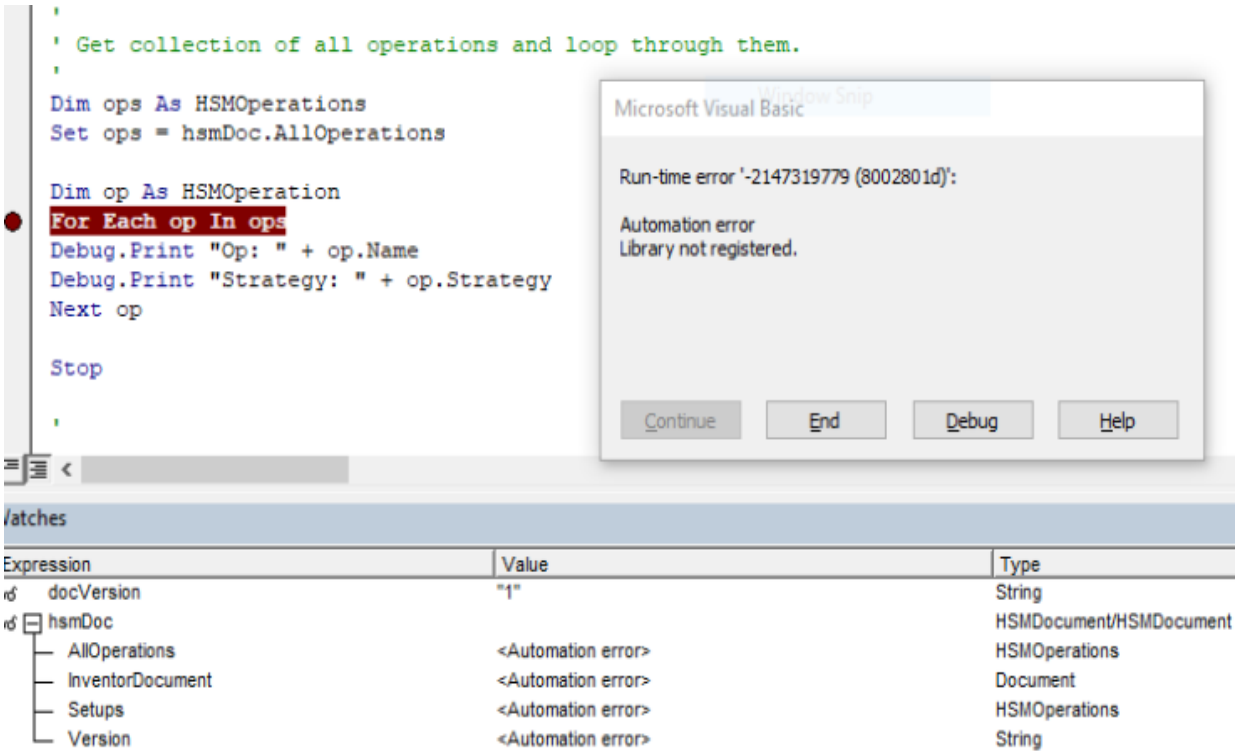
Script postprocesoru JS je rozdělen do několika sekcí. Tyto sekce jsou ve své podstatě funkce. Každá tato funkce plní jednotlivé úkony během postprocesoringu a následné generování. Veškeré tyto funkce nemusejí být předem definovány, pokud nejsou aktivní během postprocesoringu. Avšak když je během tohoto scriptu na tyto funkce odkazováno, nemohou být vyvolány pro následné generování. Script se odkazuje

na prázdnou množinu, což je neslučitelné s jeho funkčností a vygenerování požadovaných výstupů. Dojde k chybovému hlášení (Obr. **35**).



Obr. 35**: Varovné analogové okno s chybovým hlášením o absenci definice funkce.**

Při generování informací do seřizovacího listu byly při odlazování zjištěny problémy s automatickým generováním názvů operací. Původní seřizovací list je automaticky napojen na knihovny, které tyto názvy obsahují. Tyto knihovny jsou psány v anglickém jazyce a jsou překládány jednoduchým překladovým souborem psaném v jazyce JS. Pokud tyto názvy nebyly vygenerovány ani v anglickém jazyce, byla lokalizována chyba v cestě toku dat těchto informací. Tato knihovna musí být přeuložena do zdrojové složky nově vytvořené aplikace. Tento problém byl lokalizován po odzkoušení přímé cesty a pokusu o její vygenerování pomocí pouze přiloženého seřizovacího listu. Při tomto pokusu byl hlášen problém s knihovnami HSMoperations (Obr. 36).

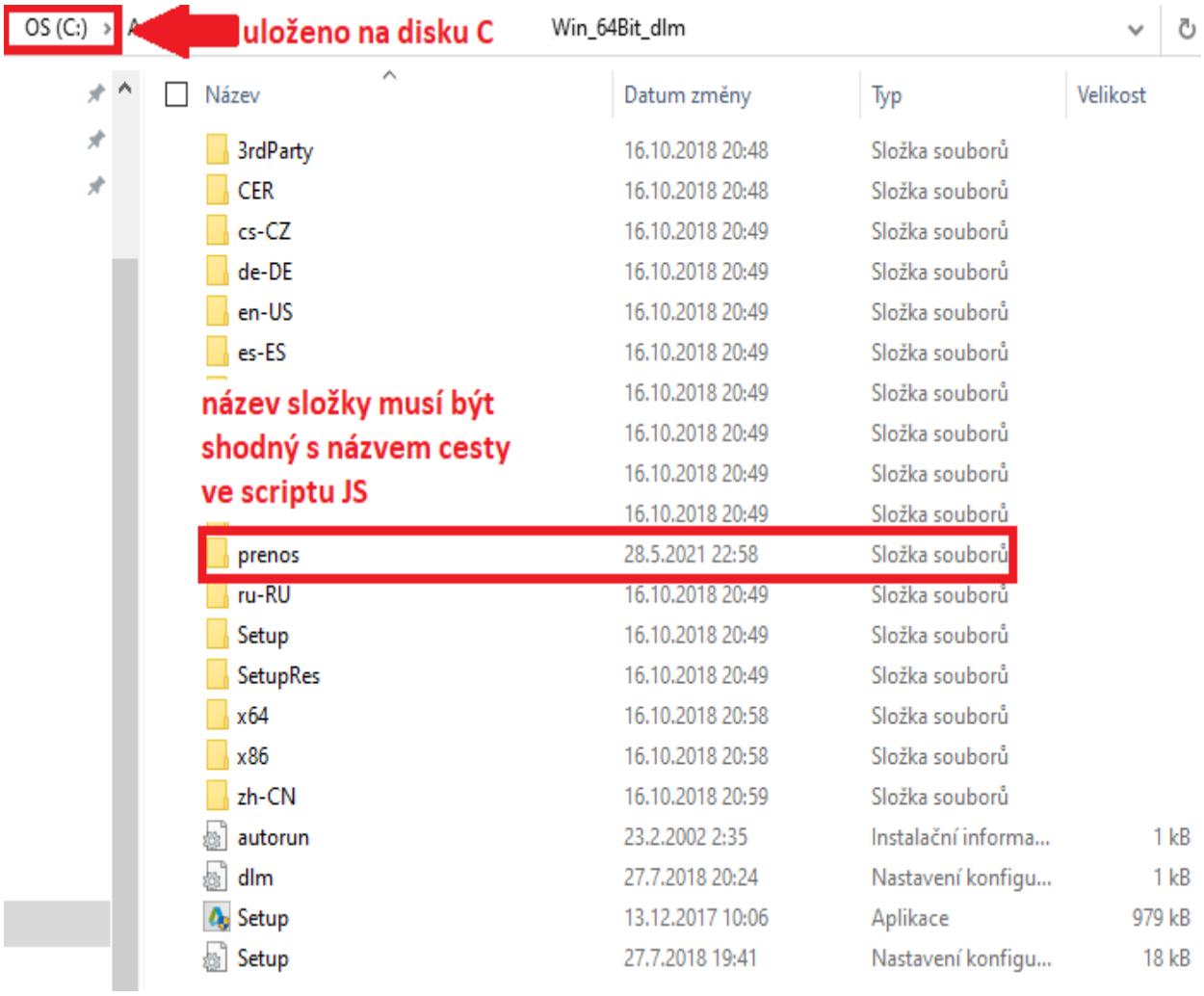


Obr. 36: Varovná hláška s chybějícími knihovnami HSMoperations.

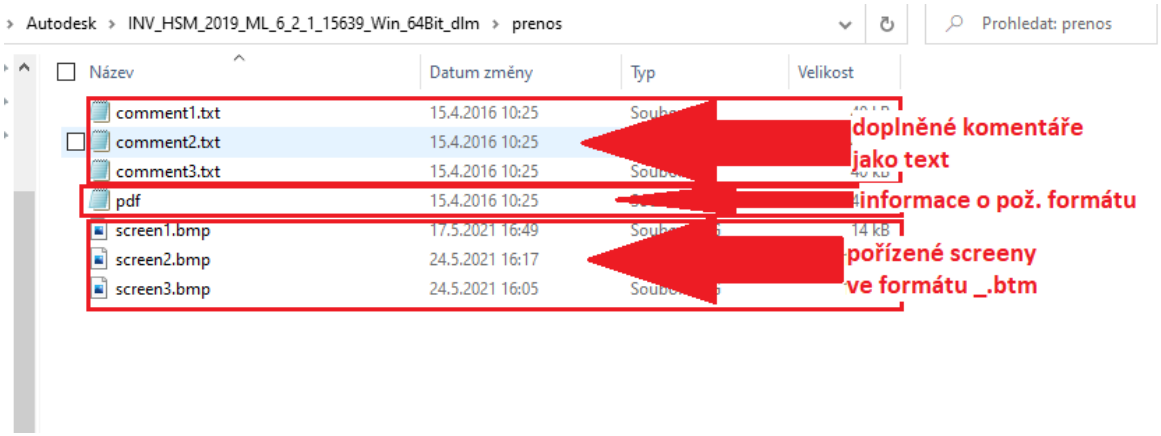
4.6 Spojení aplikace se scriptem seřizovacího listu

Jelikož je projekt řešen ve dvou rozdílných programovacích jazycích, není možné je na sebe vzájemně přímo navázat a zajistit jejich komunikaci. Rozdíl je nejen v syntaxi psaného kódu a ovladačů pro dané aplikace, ale i logické principy obou jazyků, které jsou nakompatibilní. Co ovšem zůstává stejné při práci s nimi jsou data, s kterými chceme pracovat a modifikovat je. V tomto případě to jsou data zaznamenána pomocí vytvořené konzolové aplikace DATA SHEET TOOL MANAGER. Tato data jsou nahrána do složky, která pro ně byla samostatně vytvořena. Složka v tomto případě slouží jako DATABOX, která poskytuje informace pro následné zpracování Tato složka je společná pro obě aplikace (Obr. 37). Co je ovšem rozdílné jsou směry toků těchto dat.

DATABOX IN – již z názvu plyne, že se jedná o princip naplnění tohoto databoxu potřebnými informacemi, které jsou voleny v prostředí konzolové aplikace. Do složky jsou uloženy obrázky jako bitmapy, komentáře jako samostatný text, formát jako informace pro pokyn v JS. Tyto informace jsou nutné pro následný přenos, popřípadě informaci pro aplikaci JS. Na Obr. 38 lze vidět, jakým způsobem jsou uloženy do DATABOXU. Název, pod kterým je uložen z prostředí C# je totožný s odkazem, který je v prostředí JS.



Obr. 37: Složka pro ukládání a přenos dat – DATABOX.



Obr. 38: Obsah DATABOXU.

Jelikož v této práci není možné publikovat celý Script-3, bude logika tohoto přenosu demonstrována na pořízených screenech pro následné generování do seřizovacího listu. Tento kód se nachází v prostředí scriptu JS a je integrován v sekci OnClose. Tato sekce

je pro přidané přílohy nejvhodnější, jelikož požadované přílohy jsou požadovány na konci vygenerovaného seřizovacího listu. Tyto screeny jsou ukládány jako bitmapy, to znamená ve stejném formátu jako obrázky generované samotným Inventorem. Není nutné řešit převod formátů pořízených screenů na úrovni jejich zaznamenání. Také není nutné definovat formátování daných příloh, pouze jejich umístění ve vygenerované seřizovacím listě. Lze je tedy odkazovat na data uložená z prostředí C#, aniž by musely být modifikovány do stejného formátu.

DATABOX OUT – v prostředí JS je vytvořen odkaz na infomace uložené v DATABOXU, v tomto případě nahrané bitmapy (Obr. **39**). Cílová složka a jejich názvy musejí být zachovány pro správnou a úplnou funkčnost. Tyto soubory jsou nahrány do prostředí JS, ve kterém se mohou dále modifikovat a editovat jako obrázky a texty generované z prostředí Inventoru. Není tedy nutné vytvářet vlastní formátování. To je zachováno jako v původním seřizovacím listě. Možnost umístění daných výstupů je korigováno klasickými funkcemi JS, jako například umístění obrázku pomocí funkce /align/ na Obr. **39**, v tomto případě centrovat na střed okna tabulky. Vzhled jako takový zůstane zachován dle předlohy seřizovacího listu.

```
//////////////////////////////////// Zobrazení příloh začátek
  writeln("<table class=\"sheet\" cellspacing=\"0\" align=\"center\">");

      var alignment = "center" ;   <- umístění obrázku v seřizovacím listě(uprostřed kolonky)
      var path="C:\\prenos\\screen1.bmp"   odkaz na bitmapu 1/screen 1-složka PŘENOS uložena na disku C
      if (FileSystem.isFile(path)) {
        write(makeRow("<th colspan=\"4\">Příloha1</th>"));   <- popis přílohy
        var src = getImageAsImgSrc(path);
        write("<td class=\"model\" align=\"" + alignment + "\"><img src=\"" + src + "\"/></td>");
      }

      var path="C:\\prenos\\screen2.bmp"
      if (FileSystem.isFile(path)) {
        write(makeRow("<th colspan=\"4\">Příloha2</th>"));
        var src = getImageAsImgSrc(path);
        write("<td class=\"model\" align=\"" + alignment + "\"><img src=\"" + src + "\"/></td>");
      }  SCREEN 2

      var path="C:\\prenos\\screen3.bmp"
      if (FileSystem.isFile(path)) {
        write(makeRow("<th colspan=\"4\">Příloha3</th>"));
        var src = getImageAsImgSrc(path);
        write("<td class=\"model\" align=\"" + alignment + "\"><img src=\"" + src + "\"/></td>");
      }  SCREEN 3

  writeln("</table>");
  writeln("");
//////////////////////////////////// Zobrazení příloh konec
```

HTML-generování obsahu

Obr. 39**: Ukázka propojení scriptů.**

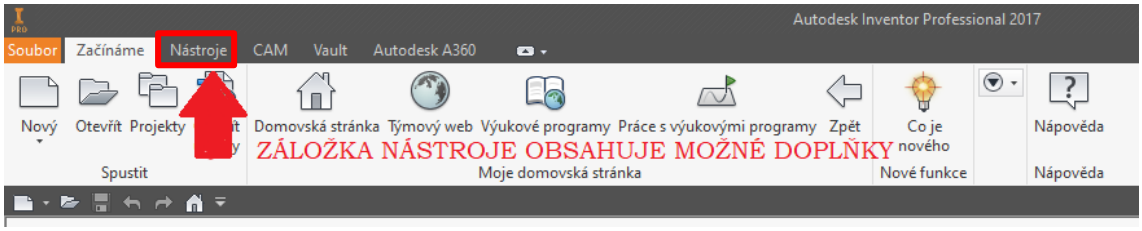
4.7 Integrace aplikace do prostředí HSM

Integrace čili začlenění, je v prostředí InventorHSM používáno od samého počátku instalace daného SW. Jde o přidávání jednotlivých aplikací, které jsou součástí kompletního balíčku Autodesk. Některé tyto aplikace jsou instalovány automaticky a jsou již součástí základního instalačního souboru Autodesk.

Nesmí být opomenuto, že i aplikace CAM je v podstatě doplněnou přidanou nadstavbou na původní Autodesk Inventor. To je řešeno při instalaci pouhým označením žádoucích nástrojů při práci s tímto SW.

Již s plně integrovaným CAM SW jsou k dispozici další nadstavby od společnosti Autodesk, které lze modifikovat a integrovat do prostředí HSM. Jako příklad lze uvést iLogic, nebo Vault. Tyto aplikace jsou vkládány přes AutodeskConfiguratorAddIn, což je systémová aplikace pro konfiguraci těchto aplikací.

Stejným způsobem bylo postupováno i s aplikací DATA SHEET TOOL MANAGER. Při spuštění Inventoru máme v základní nabídce záložku NÁSTROJE (Obr. 40). Vytvořená konzolová aplikace je integrována do prostředí HSM, tedy CAM SW, v pravé horní části. Jednoduchým kliknutím je aplikace DATA SHEET TOOL MANAGER automaticky spuštěna. Po spuštění se otevře dialogové okno s příkazy pro jednotlivé screeny a jejich komentáře. V prostředí InventorHSM lze po skrytí aplikace pracovat v klasickém režimu CAM SW. Je možné pohybovat modelem, posouvat čas simulace do požadované polohy nebo přibližovat a oddalovat obráběný díl.



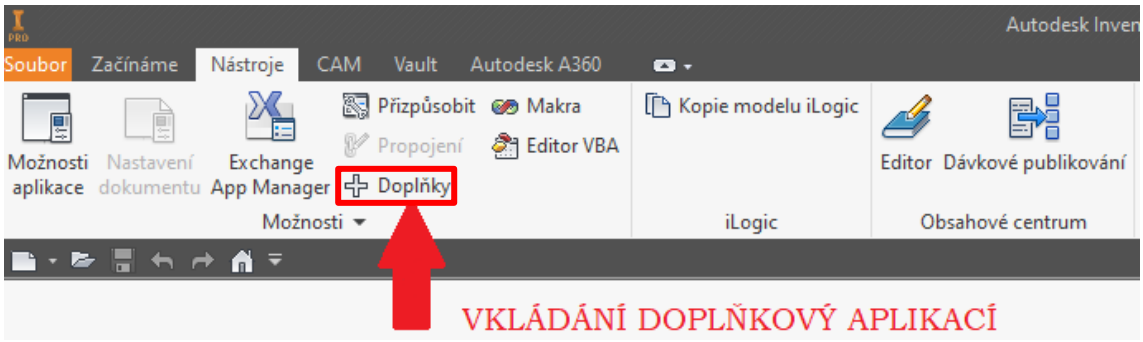
Obr. 40: Záložka NÁSTROJE v prostředí Autodesk Inventor SW.

Po otevření této záložky máme k dispozici několik funkcí. Většina z nich slouží pro editaci, konfiguraci a doplňování daného SW. Je zde k dispozici ExchangeAppManager, kterým lze spravovat veškeré aplikace a procesy v prostředí SW Inventor (Obr. 40).

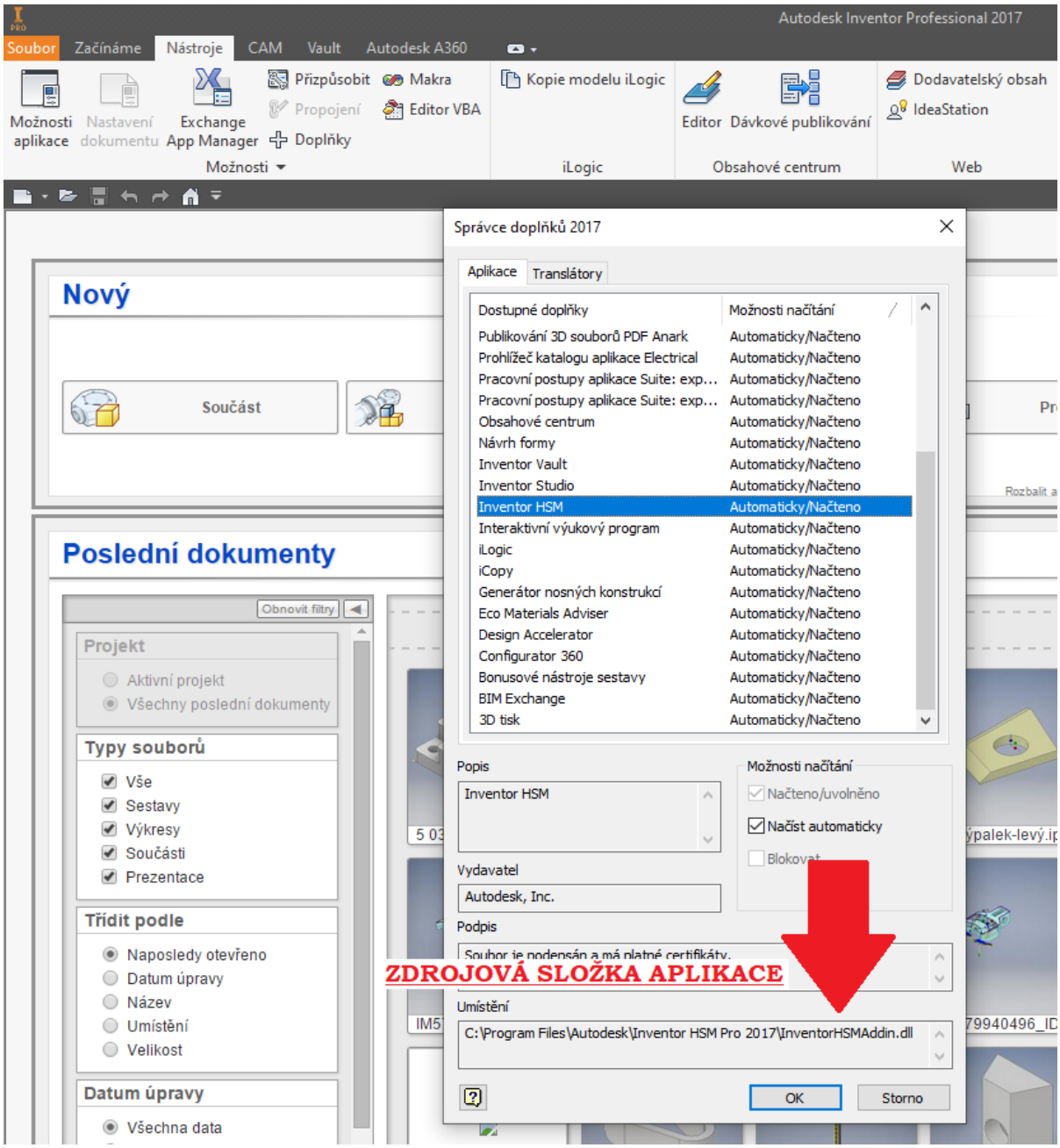
V tomto případě postačí předem definovaná funkce DOPLŇKY, pomocí níž je otevřeno okno pro vkládání nových aplikací, respektive odkazování se na zdrojové

kódy a soubory těchto aplikací (Obr. 41). Správa doplňků těchto aplikací obsahuje jejich seznam a možnosti volby jejich funkcionality. Například zda má být spuštěna automaticky, kde se bude nacházet a další případné poznámky a popisy. Pokud má být vložena nová aplikace, je nutné nejdříve uložit veškeré instalační soubory do instalačních složek Inventoru HSM, na které se bude tento příkaz odkazovat (Obr. 42). V dolní části SPRÁVCE DOPLŇKU byla zvolena složka zdrojových souborů. Díky tomu byla aplikace automaticky nahrána do horního seznamu (Obr. 43).

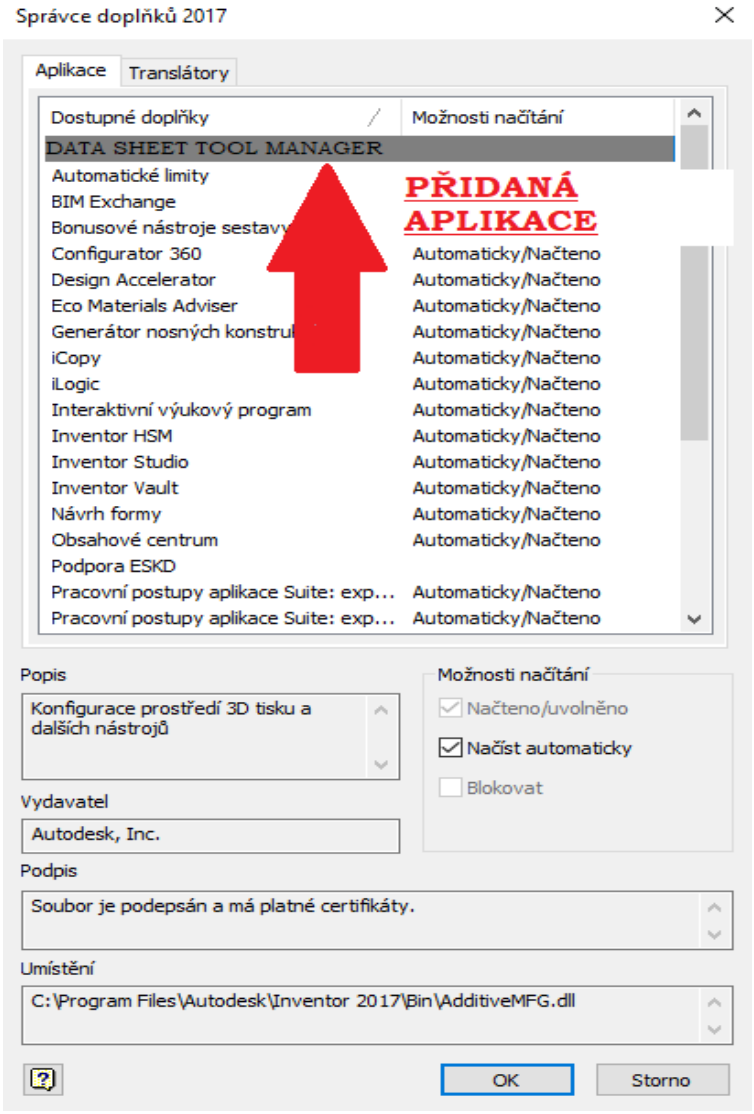
Bylo zvoleno AUTOMATICKÉ NAČÍTÁNÍ aplikace. V prostředí SW Inventor je ikona této aplikace zobrazena jako ostatní doplňkové aplikace od Autodesku, nebo jiné zvolené aplikace. V pásu karet NÁSTROJE je pravým tlačítkem otevřeno menu pro vkládání možných integrací daného SW (Obr. 44). To znamená že při spuštění CAM SW je zde aplikace připravena k okamžitému použití a její ikona je v prostředí SW CAM k dispozici (Obr. 45).



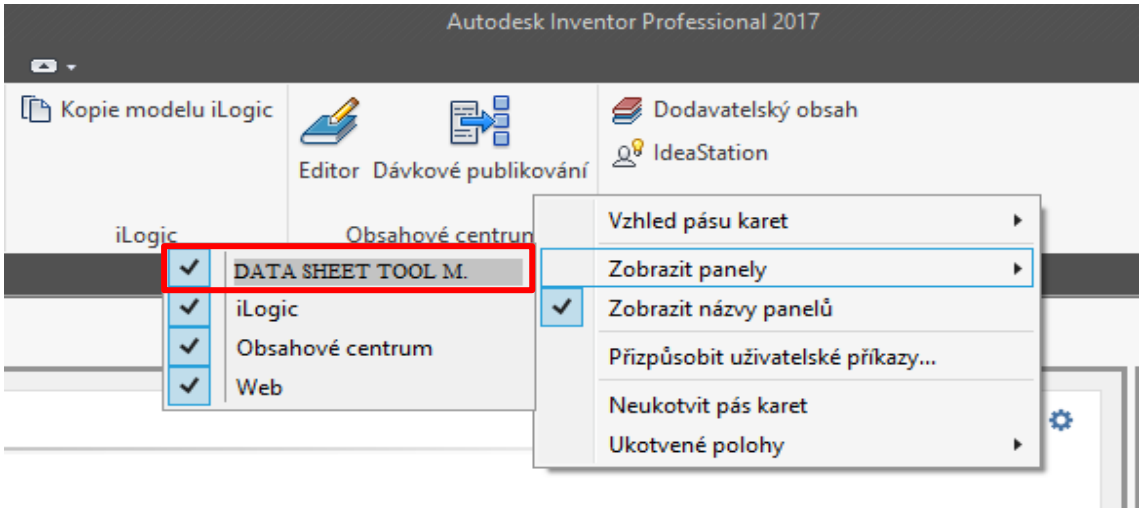
Obr. 41: Doplňky v prostředí Autodesk Inventor SW.



Obr. 42: Volba zdrojové složky aplikace v prostředí Inventor SW.



Obr. 43: Správce doplňků v prostředí Autodesk Inventor SW.



Obr. 44: Aktivace ikony v prostředí Autodesk Inventor SW.

5 Shrnutí a závěr

V první části této práce byly stanoveny cíle, které byly zadány poptávkou běžných uživatelů tohoto SW. Jelikož původní verze nebyla vyhovující a v prostředí HSM nebylo možné dosáhnout požadovaného výstupu, byl tento projekt zadán tak, aby bylo možné na něj postupně navazovat další komponenty a aplikační nadstavby. Základním cílem této práce bylo upravit stávající seřizovací list a vytvořit aplikaci, která jej doplňuje o případné obrázky během simulace. Lze konstatovat, že tohoto cíle bylo dosaženo, aplikace je plně funkční, plně integrována do prostředí HSM a je připravena k dalším případným úpravám a doplňujícím aplikačním nadstavbám.

5.1 Změna programovacího prostředí

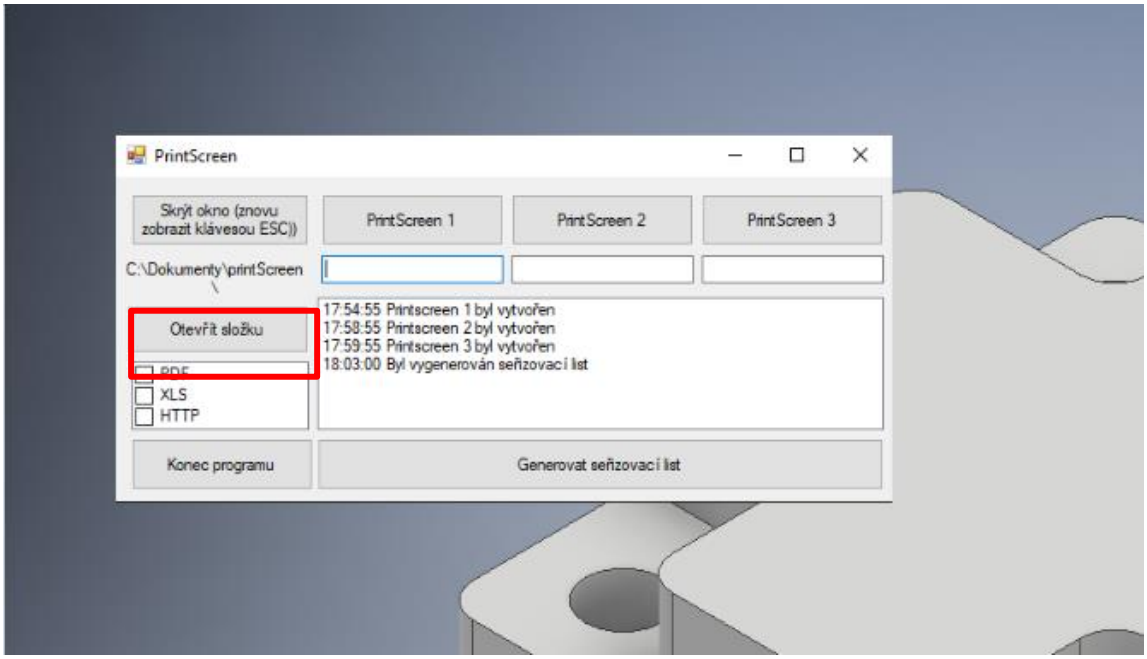
Aplikace byla vytvořena v prostředí C#, oproti původně navrhovanému prostředí JavaAPI. Důvodem byl rozsah aplikace, která byla vyvíjena samostatně. Jelikož funkci SCREEN obsahuje samotné prostředí WINDOWS, byla zvolena cesta tvorby samostatné konzolové aplikace, která byla následně navázána na seřizovací list. Použitím JavaAPI by v tomto případě bylo zbytečně složité, jejich knihovny pro tyto funkce nepotřebné a z hlediska přehlednosti nevhodné. Prostředí Visual Studio je v posledních letech značně oblíbené IDE společně s programovacím jazykem C# a je zde dobrou vizí do budoucna z hlediska jednoduchosti a modifikovatelnosti daných scriptů pro ostatní vývojáře.

5.2 Změny v konzolové aplikaci DATA SHEET TOOL MANAGER

Jelikož tato aplikace bude sloužit běžným uživatelům CAM SW, byla zde nutnost odzkoušet daný projekt v praxi. Při prvních testech bylo zaznamenáno hned několik postřehů, které během reálného procesu při editaci seřizovacího listu působily problémy uživateli daného SW.

Jako první krok, který po otevření aplikace nastane je otevření dialogového okna. Toto okno se většinou otevře uprostřed obrazovky a zabírá větší část okna pro simulaci. Při tvorbě screenů je viditelnost tohoto okna nežádoucí. Ani jeho přesouvání zde není nejlepším řešením i při použití dvou monitorů, nehledě při přepínání SW/APLIKACE. Z těchto důvodů byl dán požadavek, aby během simulace a jejího nastavení a místa zastavení dle potřeb, bylo toto okno skryté, ale zároveň stále aktivní. Vytvořením nového tlačítka SKRÝT OKNO (Obr. 45) je zde možnost dané okno schovat na pozadí a pracovat

v prostředí CAM. Tlačítkem ESC je dialogové okno znovu zobrazeno a lze dále pokračovat v prostředí aplikace.

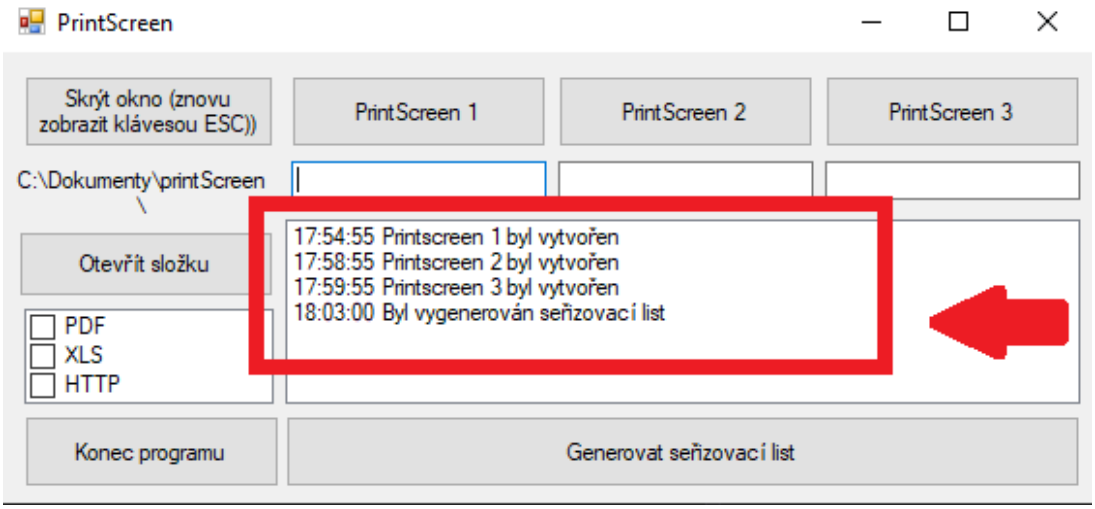


Obr. 45: Přepínání SW/APLIKACE.

5.3 Výpis o vytvoření screenů

I v případě kontroly funkčnosti aplikace a shledáním, že veškeré funkce jsou aktivní, je potřeba kontrolovat tyto procesy průběžně. Při každém úkonu by měla být zpětná vazba, že daný proces proběhl (Obr. 46).

Při tvorbě screenů je v prostředí aplikace doplněn výpisový řádek, který dává informace o provedené akci. Pokud je tedy vytvořen screen, v tomto řádku se objeví výpis, který screen byl vytvořen a v jakém čase byl vytvořen. Pokud by jeden z vytvořených screenů byl vybrán jako nevhodný, je zde možnost jeho automatického přepsání jednoduchým kliknutím na danou ikonu screenu. Ten je automaticky přeuložen a je aktivní poslední zvolená verze.



Obr. 46: Výpis o provedení akce ve Window Forms.

5.4 Pro vývojáře

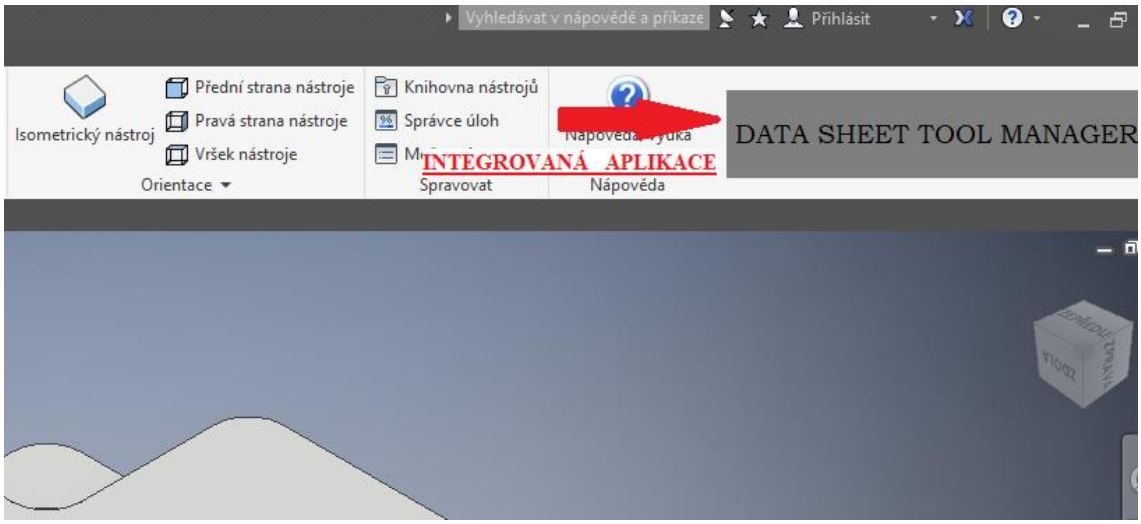
Finální výsledek tohoto projektu lze popsat na dvou úrovních – z pohledu uživatele a vývojáře.

Pro vývojáře jsou k dispozici dva scripty, které jsou na sebe navázány a jsou schopny si vzájemně předat potřebné informace pro vytvoření výstupního souboru pro uživatele. Tyto scripty jsou psány ve dvou programovacích jazycích a to JavaScript (seřizovací list) a C# (konzolová aplikace). Oba tyto soubory je možné editovat a upravovat ve svém vlastním prostředí. Je možné je dále rozšiřovat a doplňovat funkce, které by mohly sloužit pro zlepšení účelnosti výstupního souboru. Oba tyto zdrojové kódy jsou uloženy v instalačních souborech InventorHSM.

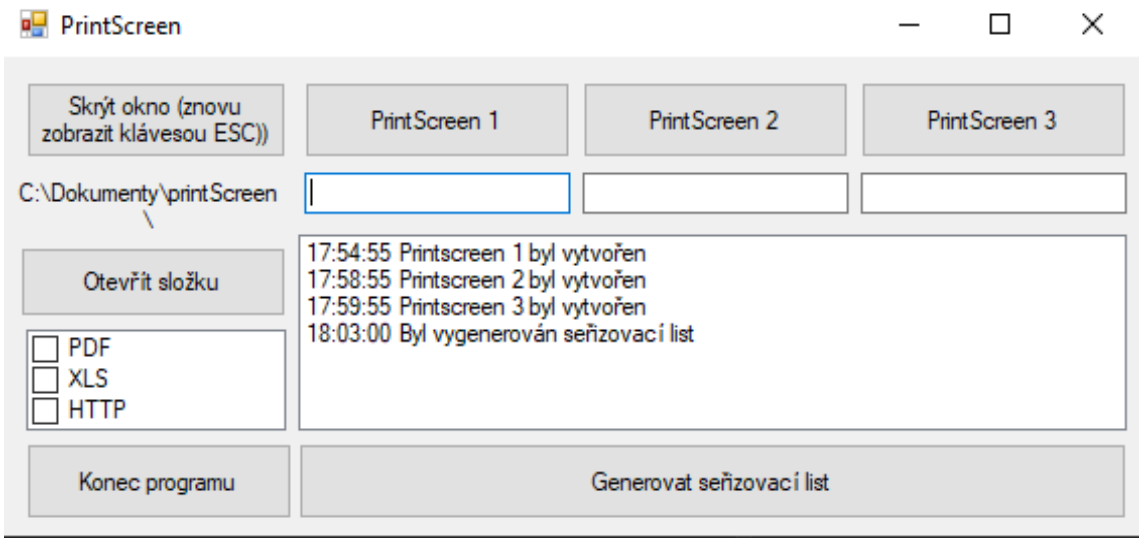
5.5 Uživatelský výstup

Konzolová aplikace je integrována do prostředí HSM a je dostupná běžnému uživateli při spuštění CAM SW. Pro běžného uživatele je tedy výstupem tohoto projektu prostředí konzolové aplikace, pomocí niž je schopen doplnit informace do seřizovacího listu. Ten je finálním výstupem při zadání a vyplnění veškerých doplňkových funkcí dané aplikace. Tento soubor lze vygenerovat ve formátu, který je možné navolit v prostředí aplikace. Momentálně je možný výstup ve formátu _.html, který je graficky vytvořen v prostředí JS. Pro formát _.pdf lze také použít prostředí JS ke grafickým úpravám.

Pro generování souborů _.xls je nutné použít souboru JS a přidané šablony formátu_.slx, pomocí které daný výstupní formát editujeme.



Obr. 47: Ikona integrované aplikace v prostředí Autodesk Inventor SW.



Obr. 48: DATA SHEET TOOL MANAGER-po spuštění.

Vytvořením tohoto projektu bylo dosaženo zpřehlednění, rozšíření a optimalizace procesu generování seřizovacího listu. Obsluha má k dispozici veškeré potřebné informace pro seřízení stroje. Každý tento seřizovací list je vygenerován pro konkrétní součást. Vystupuje jako jednotný soubor ve formátu _.pdf. Jeho účelnost je zaměřena především na přehlednost a tvorbu grafických zobrazení během simulace. Generování těchto grafických zobrazení slouží především pro představu reálného řezného procesu v problematických operacích nebo jednotlivých úkonech. Je předpokládáno, že tyto

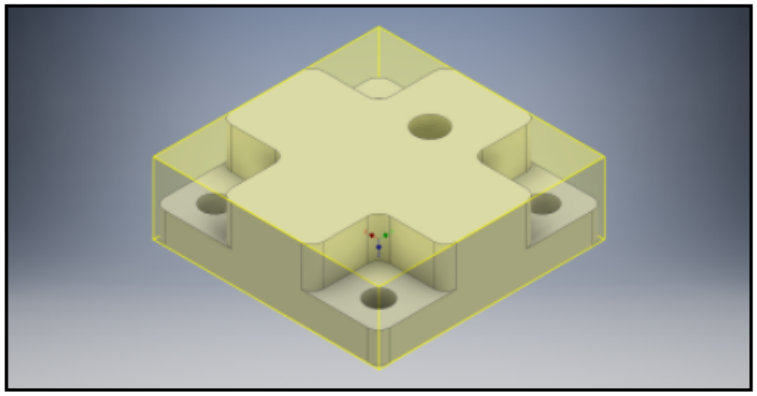
zobrazení budou často využívány při nájezdech nástroje k materiálu, především při jejich nepřehlednosti přímo ve stroji.

Přínosy aplikace:

- možnost přidání obrázků ze simulace řezného procesu,
- možnost přidání komentářů k těmto obrázkům,
- odstranění nepotřebných informací,
- eliminace rozsahu seřizovacího listu se zachováním jeho informativnosti,
- automatické označení screenu názvem operace, ve kterém byl pořízen,
- automatické vyplnění hlavičky dle přihlášeného uživatele,
- možnost dalšího rozšíření seřizovacího listu,
- zpřehlednění seřizovacího listu,
- prostor pro individuální úpravy dle požadavků uživatele,
- možnost generovat informace do seřizovacího listu z prostředí samotného inventoru,
- možnost přenosu dat v elektronické podobě,
- možnost navázat na informační zakázkové systémy ERP/PDM/PLM dle požadavků jednotlivých uživatelů nebo společností.

Projekt

Programoval:
Touš Václav
ID součásti:
ZČU 22-33-44-55-66
číslo výkresu:
001-22-33-44-55-66
Datum:
1.1.2020
Poznámka:
odlazeno 1.1.2019-Touš



Celkem

POCET OPERACI: 3
POCET NASTROJU: 3
NASTROJE: **T18 T19 T20**
MAX. POSUV: 1200mm/min
MAX. OTACKY VRETENE: 6000ot/min
ODHADOVANY CAS CYKLU: 4m:56s

Operace 1/3
POPIS: D=3-FR VHM
STRATEGIE: 2D Kontura
POCATEK: #1
PRIDAVEK: 0mm
MAX. STRANOVY KROK: 2.85mm

MAX. OTACKY VRETENE: 6000ot/min
MAX. POSUV: 300mm/min
ODHADOVANY CAS CYKLU: 57s (19.3%)
CHLAZENI: Kapalina

T20 D20 L20
TYP: válcová fréza
PRUMER: 3mm
DELKA: 30mm
BRITY: 4
DRZAK: Default Holder

Operace 2/3
POPIS: D=5,7-GUHRING RF100-DIVER
STRATEGIE: Kapsovaci
POCATEK: #1
PRIDAVEK: 0mm
MAX. STRANOVY KROK: 5.32mm

MAX. OTACKY VRETENE: 6000ot/min
MAX. POSUV: 1000mm/min
ODHADOVANY CAS CYKLU: 1m:46s (35.8%)
CHLAZENI: Kapalina

T19 D19 L19
TYP: válcová fréza
PRUMER: 5.6mm
DELKA: 30mm
BRITY: 4
DRZAK: Default Holder

Operace 3/3
POPIS: D=8-SRAZEC
STRATEGIE: 2D Kontura
POCATEK: #1
PRIDAVEK: 0mm
MAX. STRANOVY KROK: 7.6mm
KOMPENZACE: rizeni (Levy)
BEZPECNY PRUMER NASTROJE: < 12mm

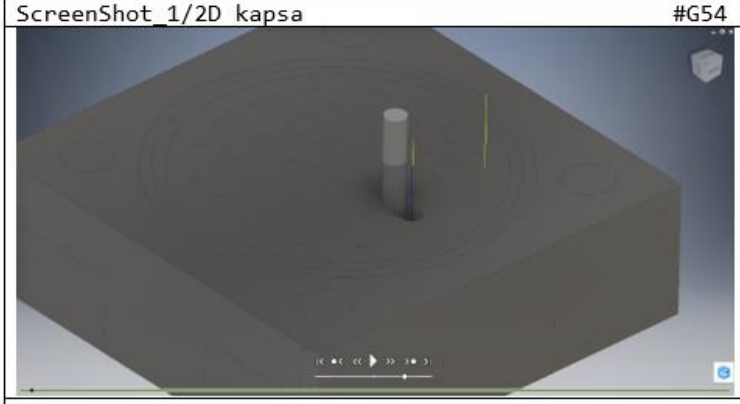
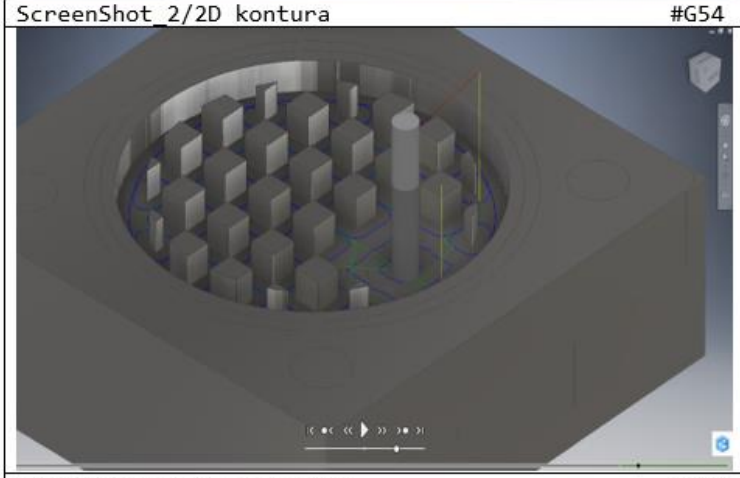
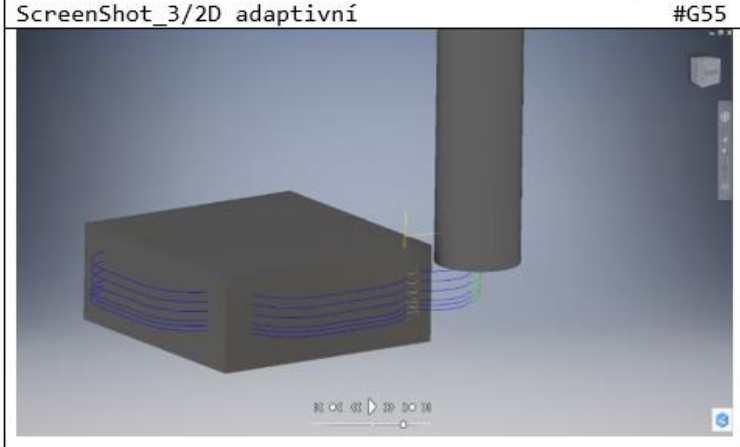
MAX. OTACKY VRETENE: 5000ot/min
MAX. POSUV: 1200mm/min
ODHADOVANY CAS CYKLU: 1m:26s (29%)
CHLAZENI: Kapalina

T18 D18 L18
TYP: úkosová fréza
PRUMER: 8mm
UHEL UKOSU: 45°
DELKA: 50mm
BRITY: 4
DRZAK: Default Holder

Obr. 49: Nový seřizovací list.

DATA SHEET TOOL MANAGER-InventorHSMpro 2021

OBSAH:
ScreenShot_1-2D kapsa
ScreenShot_2-2D kontura
ScreenShot_3-2D adaptivní

ScreenShot_1/2D kapsa #G54	Poznámka_1
	-fréza VHM se zanořuje do plného materiálu =>výkresová změna=není předvrtána díra se závitem
ScreenShot_2/2D kontura #G54	Poznámka_2
	-fréza vyjede nad materiál a zanoří se jako při prvním nájezdu=>vnitřní díra na čisto
ScreenShot_3/2D adaptivní #G55	Poznámka_3
	-nájezd do materiálu v G55

Obr. 50: DATA SHEET TOOL MANAGER – výstup.

Aplikace nejen že splňuje účel, který byl stanoven na začátku projektu, ale bude sloužit jako dobrý základ pro rozšíření na interaktivní aplikační prostředí, hlavně při práci v sestavách. Je předpokládáno, že bude rozšířena pro práci ve 3D prostředí. Využívány budou především možnosti použití upínek v sestavách, jejich zobrazení a popis

v seřizovacím listě. Tyto upínky budou vytvořeny v prostředí Autodesk Inventor a budou možné parametricky měnit v prostředí aplikace. K dispozici bude i jejich 3D náhled v prostředí aplikace. Velký důraz bude kladen na simplifikaci těchto procesů.

Vzhledem k faktu, že tato aplikace by měla být distribuována integrátorem PDM/PLM/CAD/CAM, nelze publikovat její kompletní verzi v elektronické podobě. Tato aplikace bude obsahovat vlastní licenci, kterou bude mít k dispozici oprávněný uživatel. Z tohoto důvodu jsou v této diplomové práci uvedeny scripty v psané podobě, tedy bez ovládacích a integračních prvků. Nejsou zde také zmíněny aplikační cesty daných scriptů. Script 1 a Script 2 jsou v psané formě kompletní, jelikož se jedná o úpravu základní verze seřizovacího listu JS společnosti Autodesk a tvorby aplikace obsahující funkce MS Windows. Z důvodu ochrany autorských práv společnosti, je Script 3 eliminován na veřejně publikovatelnou verzi. V tomto scriptu je již zpracován vlastní postup a vývoj dané aplikace. Script 3 není publikován kompletní, avšak je zde vypsán v takovém rozsahu, aby byla zřejmá logika daných funkcí a obecná funkcionalita scriptu.

Seznam literatury a informačních zdrojů

[1] JANDEČKA, K., ČESÁNEK, J., KOŽMÍN, P. (2000). Programování NC strojů. Plzeň ZČU. ISBN 80-7082-692-4

[2] NÁPRSTKOVÁ, N., JANDEČKA, K. (2010). Programování výrobních strojů. ústí nad Labem, Univerzita J.E. Purkyně. ISBN 97-8807-414-2161

[3] SCHILDT, H. (2005). Jawa: the complete reference, McGraw-Hill/osborne. ISBN 00-7223-073-8

[4] STANĚK, J., NĚMEC, J. (2005). Metodika zpracování a úprava diplomových prací, Plzeň: ZČU.

[5] API – Multimediaexpo.cz. Multimediální česká otevřená encyklopedie – Multimediaexpo.cz [online]. Dostupné z: http://www.multimediaexpo.cz/mmecz/index.php/API

[6] Co je to API a jaké jsou možnosti jeho využití. [online]. Copyright © 2021 Rascasone s.r.o. [cit. 06.05.2021]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-api

[7] JavaScript Tutorial. W3Schools Online Web Tutorials [online]. Dostupné z: https://www.w3schools.com/js/default.asp

[8] Proměnné v JavaScriptu | Interval.cz. Interval.cz | Svět Internetu, Technologií a Bezpečnosti [online]. Copyright © [cit. 06.05.2021]. Dostupné z: https://www.interval.cz/clanky/promenne-v-javascriptu/

[9] Co je to API (application programming interface)? Definice pojmu. Co je SEO? Váš web v TOP 1 ve vyhledávačích a nárůst online prodejů [online]. Dostupné z: https://topranker.cz/slovnik/co-je-to-api-application-programming-interface/

[10] JavaScript.com | Functions. JavaScript.com [online]. Copyright © 2016 [cit. 06.05.2021]. Dostupné z: https://www.javascript.com/learn/functions

[11] JavaScript.com | Operators. JavaScript.com [online]. Copyright © 2016 [cit. 07.05.2021]. Dostupné z: https://www.javascript.com/learn/operators

[12] Blog - Bim solutions. Home - Bim solutions [online]. Dostupné z: https://www.solutions-tcc.com/blogs/

[13] Komarek System. [online]. Copyright © [cit. 08.05.2021]. Dostupné z: https://cz.komareksystem-app.com/?session=3b1b8b9c37924cf09909ccdd63a3874f&aff_id=9932&fpp=1&pixelsettings=dijaqaqa.zeretu.xyz%2Ffbp%3Fev%3D%7Bev%7D%26pixel%3D%7Bpixel%7D

[14] GitHub - Autodesk/cam-posteditor: Autodesk HSM Post Processor for Visual Studio Code. GitHub: Where the world builds software · GitHub [online]. Copyright © 2021 GitHub, Inc. [cit. 09.05.2021]. Dostupné z: https://github.com/Autodesk/cam-posteditor

[15] Weekly HSM Fast Track Hangout - API Programming - YouTube. YouTube [online]. Copyright © 2021 Google LLC [cit. 25.05.2021]. Dostupné z: https://www.youtube.com/watch?v=yiRSybi6Sak

[16] JavaScript.com | Variables. JavaScript.com [online]. Copyright © 2016 [cit. 10.05.2021]. Dostupné z: https://www.javascript.com/learn/variables

[17] VIRIUS, Miroslav. (2021). Programování v C#: od základů k profesionálnímu použití. Praha: Grada Publishing. Knihovna programátora (Grada). ISBN 978-80-271-1216-6.

[18] Online kurzy programování C#.NET - Největší český e-learning. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2021 itnetwork.cz. Veškerý obsah webu [cit. 05.05.2021]. Dostupné z: https://www.itnetwork.cz/csharp

[19] DAVIM, J. PAULO: Modern machining technology: a practical guide: Oxford, Woodhead Publishing 2011, ISBN 978-0-85709-099-7

Seznam obrázků

Seznam tabulek

Seznam příloh

Přílohy

Příloha 1: Script DATA SHEET

```
description = "Setup Sheet";
vendor = "Autodesk";
vendorUrl = "http://www.autodesk.com";
legal = "Copyright (C) 2012-2015 by Autodesk, Inc.";
certificationLevel = 2;


longDescription = "Setup sheet for generating an HTML document with the relevant details for the setup, tools, and individual operations. You can print the document directly or alternatively convert it to a PDF file for later reference.";


capabilities = CAPABILITY_SETUP_SHEET;
extension = "html";
mimetype = "text/html";
keywords = "MODEL_IMAGE PREVIEW_IMAGE";
setCodePage("utf-8");
dependencies = "setup-sheet.css";


allowMachineChangeOnSection = true;


properties = {
  embedStylesheet: true, // embeds the stylesheet
  useUnitSymbol: false, // specifies that symbols should be used for units (some printers may not support this)
  showDocumentPath: true, // specifies that the path of the source document should be shown
  showModelImage: true, // specifies that the model image should be shown
  showToolImage: true, // specifies that the tool image should be shown
  showPreviewImage: true, // specifies that the preview image should be shown
  previewWidth: "8cm", // the width of the preview picture
  showPercentages: true, // specifies that the percentage of the total cycle time should be shown for each operation cycle time
  showFooter: true, // specifies that the footer should be shown
  showRapidDistance: true,
  rapidFeed: 5000, // the rapid traversal feed
  toolChangeTime: 15, // the time in seconds for a tool change
  showNotes: true, // show notes for the operations
  forcePreview: false, // enable to force preview picture for all pattern instances
  showOperations: true, // enable to see information for each operation
  showTools: false, // enable to see information for each tools
```

```
  showTotals: true // enable to see total information
};

var useToolNumber = true;

var feedFormat = createFormat({decimals:(unit == MM ? 1 : 3)});
var toolFormat = createFormat({decimals:0});
var rpmFormat = createFormat({decimals:0});
var secFormat = createFormat({decimals:3});
var angleFormat = createFormat({decimals:0, scale:DEG});
var pitchFormat = createFormat({decimals:3});
var spatialFormat = createFormat({decimals:(unit == MM ? 2 : 3)});
var percentageFormat = createFormat({decimals:1, scale:100});
var timeFormat = createFormat({decimals:2});
var taperFormat = angleFormat; // share format

// collected state
var zRanges = {};
var totalCycleTime = 0;
var exportedTools = {};
var toolRenderer;

function getUnitSymbolAsString() {
  switch (unit) {
  case MM:
    return properties.useUnitSymbol ? "&#x339c;" : "mm";
  case IN:
    return properties.useUnitSymbol ? "&#x2233;" : "in";
  default:
    error(localize("Unit is not supported."));
    return undefined;
  }
}

function getFeedSymbolAsString() {
  switch (unit) {
  case MM:
    return properties.useUnitSymbol ? "&#x339c;/min" : "mm/min";
  case IN:
    return properties.useUnitSymbol  ? "&#x2233;/min" : "in/min";
    // return properties.useUnitSymbol  ? "&#x2032;/min" : "ft/min";
  default:
```

```
    error(localize("Unit is not supported."));
    return undefined;
  }
}

function getFPRSymbolAsString() {
  switch (unit) {
  case MM:
    return properties.useUnitSymbol ? "&#x339c;" : "mm";
  case IN:
    return properties.useUnitSymbol  ? "&#x2233;" : "in";
  default:
    error(localize("Unit is not supported."));
    return undefined;
  }
}

function toString(value) {
  if (typeof(value) == 'string') {
    return "'" + value + "'";
  } else {
    return value;
  }
}

function makeRow(content, classId) {
  if (classId) {
    return "<tr class=\"" + classId + "\">" + content + "</tr>";
  } else {
    return "<tr>" + content + "</tr>";
  }
}



function makeHeading(content, classId) {
  if (classId) {
    return "<th class=\"" + classId + "\">" + content + "</th>";
  } else {
    return "<th>" + content + "</th>";
  }
}
```

```
function makeColumn(content, classId) {
  if (classId) {
    return "<td class=\"" + classId + "\">" + content + "</td>";
  } else {
    return "<td>" + content + "</td>";
  }
}



function bold(content, classId) {
  if (classId) {
    return "<b class=\"" + classId + "\">" + content + "</b>";
  } else {
    return "<b>" + content + "</b>";
  }
}

function d(content) {
  return "<div class=\"description\" style=\"display: inline;\">" + content + "</div>";
}

function v(content) {
  return "<div class=\"value\" style=\"display: inline;\">" + content + "</div>";
}

function p(content, classId) {
  if (classId) {
    return "<p class=\"value\">" + content + "</p>";
  } else {
    return "<p>" + content + "</p>";
  }
}

var cachedParameters = {};
var patternIds = {};
var seenPatternIds = {};

function formatPatternId(id) {
  var chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  var result = "";
  while (id >= 0) {
    result = result + chars.charAt(id % chars.length);
```

```
    id -= chars.length;
  }
  return result;
}

function onParameter(name, value) {
  cachedParameters[name] = value;
}

function onOpen() {
  cachedParameters = {};

  toolRenderer = createToolRenderer();
  if (toolRenderer) {
    toolRenderer.setBackgroundColor(new Color(1, 1, 1));
    toolRenderer.setFluteColor(new Color(40.0/255, 40.0/255, 40.0/255));
    toolRenderer.setShoulderColor(new Color(80.0/255, 80.0/255, 80.0/255));
    toolRenderer.setShaftColor(new Color(80.0/255, 80.0/255, 80.0/255));
    toolRenderer.setHolderColor(new Color(40.0/255, 40.0/255, 40.0/255));
  }

  if (is3D()) {
    var numberOfSections = getNumberOfSections();
    for (var i = 0; i < numberOfSections; ++i) {
      var section = getSection(i);
      var zRange = section.getGlobalZRange();
      var tool = section.getTool();
      if (zRanges[tool.number]) {
        zRanges[tool.number].expandToRange(zRange);
      } else {
        zRanges[tool.number] = zRange;
      }
    }
  }

  write(
    "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"\n" +
    "                 \"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd\">\n"
  );
  write("<html>");
  // header
  var c = "<head>";
```

```
  c += "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">";
  if (properties.embedStylesheet) {
    c += "<style type=\"text/css\">" + loadText("setup-sheet.css", "utf-8") +
"</style>";
  } else {
    c += "<link rel=\"StyleSheet\" href=\"setup-sheet.css\" type=\"text/css\"
media=\"print, screen\">";
  }
  c += "<style type=\"text/css\">" + ".preview img {width: " + properties.previewWidth
+ ";}" + "</style>";
  if (programName) {
    c += "<title>" + localize("Setup Sheet for Program") + " " + programName +
"</title>";
  } else {
    c += "<title>" + localize("Setup Sheet") + "</title>";
  }
  c += "</head>";
  write(c);

  write("<body>");
  if (programName) {
    write("<h1>" + localize("Setup Sheet for Program") + " " + programName + "</h1>");
  } else {
    write("<h1>" + localize("Setup Sheet") + "</h1>");
  }

  patternIds = {};
  var numberOfSections = getNumberOfSections();
  var j = 0;
  for (var i = 0; i < numberOfSections; ++i) {
    var section = getSection(i);
    if (section.isPatterned()) {
      var id = section.getPatternId();
      if (patternIds[id] == undefined) {
        patternIds[id] = formatPatternId(j);
        ++j;
      }
    }
  }
}

/**
  Returns the specified coolant as a string.
```

```
*/
function getCoolantDescription(coolant) {
  switch (coolant) {
  case COOLANT_OFF:
    return localize("Off");
  case COOLANT_FLOOD:
    return localize("Flood");
  case COOLANT_MIST:
    return localize("Mist");
  case COOLANT_THROUGH_TOOL:
    return localize("Through tool");
  case COOLANT_AIR:
    return localize("Air");
  case COOLANT_AIR_THROUGH_TOOL:
    return localize("Air through tool");
  case COOLANT_SUCTION:
    return localize("Suction");
  case COOLANT_FLOOD_MIST:
    return localize("Flood and mist");
  default:
    return localize("Unknown");
  }
}

/** Formats WCS to text. */
function formatWCS(id) {
  /*
  if (id == 0) {
    id = 1;
  }
  if (id > 6) {
    return "G54.1P" + (id - 6);
  }
  return "G" + (getAsInt(id) + 53);
  */
  return "#" + id;
}

function onSection() {
  skipRemainingSection();
}
```

```
function pageWidthFitPath(path) {
  var PAGE_WIDTH = 70;
  if (path.length<PAGE_WIDTH) {
    return path;
  }
  var newPath = "";
  var tempPath = "";
  var flushPath = "";
  var offset = 0;
  var ids = "";
  for (var i = 0; i < path.length; i++) {
    var cv = path[i];
    if (i > PAGE_WIDTH + offset) {
      if (flushPath.length == 0) { // No good place to break
        flushPath = tempPath;
        tempPath = "";
      }
      newPath += flushPath + "<br/>";
      offset += flushPath.length - 1;
      flushPath = "";
    }
    if (cv == "\\" || cv == "/" || cv == " " || cv=="_") {
      flushPath += tempPath + cv;
      tempPath = "";
    } else {
      tempPath += cv;
    }
  }
  newPath += flushPath + tempPath;
  return newPath;
}


function writeTools() {
  writeln("<table class=\"sheet\" cellspacing=\"0\" align=\"center\">");
  var colgroup = "<colgroup span=\"3\"><col width=\"1*\"/><col width=\"1*\"/><col
width=\"120\"/></colgroup>";
  write(colgroup);
  write(makeRow("<th colspan=\"3\">" + localize("Tools") + "</th>"));


  var tools = getToolTable();
  if (tools.getNumberOfTools() > 0) {
```

```
    var numberOfTools = useToolNumber ? tools.getNumberOfTools() : getNumberOfSections();
    for (var i = 0; i < numberOfTools; ++i) {
      var tool = useToolNumber ? tools.getTool(i) : getSection(i).getTool();

      var c1 = "<table class=\"info\">";
      c1 += makeRow(
        makeColumn(
          bold(localize("T") + toolFormat.format(tool.number)) + " " +
          localize("D") + toolFormat.format(tool.diameterOffset) + " " +
          localize("L") + toolFormat.format(tool.lengthOffset)
        )
      );
      c1 += makeRow(makeColumn(d(localize("Type") + ": ") + v(getToolTypeName(tool.type))));
      c1 += makeRow(makeColumn(d(localize("Diameter") + ": ") + v(spatialFormat.format(tool.diameter) + getUnitSymbolAsString())));
      if (tool.cornerRadius) {
        c1 += makeRow(makeColumn(d(localize("Corner Radius") + ": ") + v(spatialFormat.format(tool.cornerRadius) + getUnitSymbolAsString())));
      }
      if ((tool.taperAngle > 0) && (tool.taperAngle < Math.PI)) {
        if (tool.isDrill()) {
          c1 += makeRow(makeColumn(d(localize("Tip Angle") + ": ") + v(taperFormat.format(tool.taperAngle) + "&deg;")));
        } else {
          c1 += makeRow(makeColumn(d(localize("Taper Angle") + ": ") + v(taperFormat.format(tool.taperAngle) + "&deg;")));
        }
      }
      c1 += makeRow(makeColumn(d(localize("Length") + ": ") + v(spatialFormat.format(tool.bodyLength) + getUnitSymbolAsString())));
      c1 += makeRow(makeColumn(d(localize("Flutes") + ": ") + v(tool.numberOfFlutes)));
      if (tool.material) {
        c1 += makeRow(makeColumn(d(localize("Material") + ": ") + v(getMaterialName(tool.material))));
      }
      if (tool.comment) {
        c1 += makeRow(makeColumn(d(localize("Description") + ": ") + v(tool.comment)));
      }
      if (tool.vendor) {
        c1 += makeRow(makeColumn(d(localize("Vendor") + ": ") + v(tool.vendor)));
      }
      //c1 += "<tr class=\"thin\"><td width=\"6cm\"> </td></tr>"; // fixed width
```

```
      c1 += "</table>";


      var c2 = "<table class=\"tool\">";
      c2 += makeRow(makeColumn(" ")); // move 1 row down
      if (zRanges[tool.number]) {
    //          c2  +=  makeRow(makeColumn(d(localize("Minimum  Z")  +  ":  ")  +
v(spatialFormat.format(zRanges[tool.number].getMinimum())                          +
getUnitSymbolAsString())));
      }


      var maximumFeed = 0;
      var maximumSpindleSpeed = 0;
      var cuttingDistance = 0;
      var rapidDistance = 0;
      var cycleTime = 0;
      for (var j = 0; j < getNumberOfSections(); ++j) {
        var section = getSection(j);
        if (section.getTool().number == tool.number) {
          maximumFeed = Math.max(maximumFeed, section.getMaximumFeedrate());
          maximumSpindleSpeed             =             Math.max(maximumSpindleSpeed,
section.getMaximumSpindleSpeed());
          cuttingDistance += section.getCuttingDistance();
          rapidDistance += section.getRapidDistance();
          cycleTime += section.getCycleTime();
        }
      }
/*
      if (properties.rapidFeed > 0) {
        cycleTime += rapidDistance/properties.rapidFeed * 60;
      }
*/
      c2    +=    makeRow(makeColumn(d(localize("Maximum    Feed")    +    ":    ")    +
v(feedFormat.format(maximumFeed) + getFeedSymbolAsString())));
      c2  +=  makeRow(makeColumn(d(localize("Maximum  Spindle  Speed")  +  ":  ")  +
v(rpmFormat.format(maximumSpindleSpeed) + localize("rpm"))));
     //  c2  +=  makeRow(makeColumn(d(localize("Cutting  Distance")  +  ":  ")  +
v(spatialFormat.format(cuttingDistance) + getUnitSymbolAsString())));
      if (properties.showRapidDistance) {
    //        c2  +=  makeRow(makeColumn(d(localize("Rapid  Distance")  +  ":  ")  +
v(spatialFormat.format(rapidDistance) + getUnitSymbolAsString())));
      }


      var additional = "";
      if ((getNumberOfSections() > 1) && properties.showPercentages) {
```

```
        if (totalCycleTime > 0) {
          additional = "<div class=\"percentage\">(" +
percentageFormat.format(cycleTime/totalCycleTime) + "%)</div>";
        }
      }
      c2 += makeRow(makeColumn(d(localize("Estimated Cycle Time") + ": ") +
v(formatCycleTime(cycleTime) + " " + additional)));
      //c2 += "<tr class=\"thin\"><td width=\"6cm\"> </td></tr>"; // fixed width
      c2 += "</table>";

      var c3 = "";
      if (toolRenderer && properties.showToolImage) {
        var id = useToolNumber ? tool.number : (i + 1);
        var path = "tool" + id + ".png";
        var width = 2.5 * 100;
        var height = 2.5 * 133;
        try {
          if (!exportedTools[id]) {
            toolRenderer.exportAs(path, "image/png", tool, width, height);
            exportedTools[id] = true; // do not export twice
          }
          c3 = "<table class=\"info\" cellspacing=\"0\">" +
            makeRow("<td class=\"image\"><img width=\"100px\" src=\"" + path +
"\"/></td>") +
            "</table>";
        } catch(e) {
        }
      }
      writeln("");

      write(
        makeRow(
          "<td valign=\"top\">" + c1 + "</td>" +
          "<td valign=\"top\">" + c2 + "</td>" +
          "<td class=\"image\" align=\"right\">" + c3 + "</td>",
          "info"
        )
      );
      if ((i + 1) < tools.getNumberOfTools()) {
        write("<tr
class=\"space\"><td> </td><td> </td><td> </td></tr>");
      }
      writeln("");
```

```
      writeln("");
    }
  }

  writeln("</table>");
  writeln("");
}

function onSectionEnd() {
  if (isFirstSection()) {
    var c = "";

    if (programComment) {
      c += makeRow(makeColumn(d(localize("Program Comment") + ": ") +
v(programComment)));
    }

    if (hasParameter("job-description")) {
      var description = getParameter("job-description");
      if (description) {
        c += makeRow(makeColumn(d(localize("Job Description") + ": ") +
v(description)));
      }
    }

    if (hasParameter("iso9000/document-control")) {
      var id = getParameter("iso9000/document-control");
      if (id) {
        c += makeRow(makeColumn(d(localize("Job ISO-9000 Control") + ": ") + v(id)));
      }
    }

    if (properties.showDocumentPath) {
      if (hasParameter("document-path")) {
        var path = getParameter("document-path");
        if (path) {
          c += makeRow(makeColumn(d(localize("Document Path") + ": ") +
v(pageWidthFitPath(path))));
        }
      }

      if (hasParameter("document-version")) {
        var version = getParameter("document-version");
```

```
        if (version) {
          c += makeRow(makeColumn(d(localize("Document Version") + ": ") +
v(version)));
        }
      }
    }

    if (properties.showNotes && hasParameter("job-notes")) {
      var notes = getParameter("job-notes");
      if (notes) {
        c +=
          "<tr class=\"notes\"><td valign=\"top\">" +
          d(localize("Notes")) + ": <pre>" + getParameter("job-notes") +
          "</pre></td></tr>";
      }
    }

    if (c) {
      write("<table class=\"jobhead\" align=\"center\">" + c + "</table>");
      write("<br>");
      writeln("");
      writeln("");
    }

    var workpiece = getWorkpiece();
    var delta = Vector.diff(workpiece.upper, workpiece.lower);
    if (delta.isNonZero() || modelImagePath && properties.showModelImage) {

      write("<table class=\"job\" cellspacing=\"0\" align=\"center\">");
      write(makeRow("<th colspan=\"2\">" + localize("Job") + "</th>"));
      write("<tr>");

      var numberOfColumns = 0;
      { // stock - workpiece
        if (delta.isNonZero()) {
          var c = "<table class=\"info\" cellspacing=\"0\">";

          var workOffset = undefined;
          var multipleWCS = false;
          var numberOfSections = getNumberOfSections();
          var workOffsets = [];
          for (var i = 0; i < numberOfSections; ++i) {
```

```
            var section = getSection(i);
            if (!workOffsets[section.workOffset]) {
              workOffsets[section.workOffset] = true;
            }
            if (!workOffset) {
              workOffset = section.workOffset;
            }
            if (workOffset != section.workOffset) {
              multipleWCS = true;
            }
          }
          var text = "";
          for (var id in workOffsets) {
            text += " " + formatWCS(id);
          }
          c += makeRow(makeColumn(d(localize("WCS")) + ":" + text));


          if (multipleWCS) {
            c += makeRow(makeColumn(d(localize("Program uses multiple WCS!"))));
          }


          c += makeRow(makeColumn(
            d(localize("Stock")) + ": <br>  " + v(localize("DX") + ": " +
spatialFormat.format(delta.x) + getUnitSymbolAsString()) + "<br>  " +
v(localize("DY") + ": " + spatialFormat.format(delta.y) + getUnitSymbolAsString()) +
"<br>  " + v(localize("DZ") + ": " + spatialFormat.format(delta.z) +
getUnitSymbolAsString())
          ));
/*
          if (hasParameter("part-lower-x") && hasParameter("part-lower-y") &&
hasParameter("part-lower-z") &&
              hasParameter("part-upper-x") && hasParameter("part-upper-y") &&
hasParameter("part-upper-z")) {
            var lower = new Vector(getParameter("part-lower-x"), getParameter("part-
lower-y"), getParameter("part-lower-z"));
            var upper = new Vector(getParameter("part-upper-x"), getParameter("part-
upper-y"), getParameter("part-upper-z"));
            var delta = Vector.diff(upper, lower);
            c += makeRow(makeColumn(
              d(localize("Part")) + ": <br>  " + v(localize("DX") + ": " +
spatialFormat.format(delta.x) + getUnitSymbolAsString() + "<br>  " +
v(localize("DY") + ": " + spatialFormat.format(delta.y) + getUnitSymbolAsString()) +
"<br>  " + v(localize("DZ") + ": " + spatialFormat.format(delta.z) +
getUnitSymbolAsString()))
            ));
          }
```

```
          c += makeRow(makeColumn(
            d(localize("Stock Lower in WCS") + " " + formatWCS(workOffset)) + ":
<br>  "   +   v("X:   "   +   spatialFormat.format(workpiece.lower.x)   +
getUnitSymbolAsString())     +     "<br>  "     +     v("Y:     "     +
spatialFormat.format(workpiece.lower.y)       +       getUnitSymbolAsString())       +
"<br>  "   +   v("Z:   "   +   spatialFormat.format(workpiece.lower.z)   +
getUnitSymbolAsString())
          ));
          c += makeRow(makeColumn(
            d(localize("Stock Upper in WCS") + " " + formatWCS(workOffset)) + ":
<br>  "   +   v("X:   "   +   spatialFormat.format(workpiece.upper.x)   +
getUnitSymbolAsString())     +     "<br>  "     +     v("Y:     "     +
spatialFormat.format(workpiece.upper.y)       +       getUnitSymbolAsString())       +
"<br>  "   +   v("Z:   "   +   spatialFormat.format(workpiece.upper.z)   +
getUnitSymbolAsString())
          ));
*/
          c += "</table>";
          write(makeColumn(c));
          ++numberOfColumns;
        }
      }

      if (modelImagePath && properties.showModelImage) {
        var                                  path                                  =
FileSystem.getCombinedPath(FileSystem.getFolderPath(getOutputPath()), modelImagePath);
        if (!FileSystem.isFile(path)) {
          warning(subst(localize("Model image doesn't exist '%1'."), path));
        }

        ++numberOfColumns;
        var alignment = (numberOfColumns <= 1) ? "center" : "right";
        write("<td  class=\"model\"  align=\""  +  alignment  +  "\"><img  src=\""  +
modelImagePath + "\"/></td>");
      }

      write("</tr>");
      write("</table>");
      write("<br>");
      writeln("");
      writeln("");
    }

    if (properties.showTotals) {
      writeTotals();
```

```
      write("<br>");
      writeln("");
      writeln("");
    }

    if (properties.showTools) {
      writeTools();
      write("<br>");
      writeln("");
      writeln("");
    }
  }

  if (!properties.showOperations) {
    return; // skip
  }

  if (isFirstSection()) {
    writeln("<table class=\"sheet\" cellspacing=\"0\" align=\"center\">");
  }

  var c1 = "<table class=\"info\" cellspacing=\"0\">";

  c1 += makeRow(
    makeColumn(v(localize("Operation") + " " + (currentSection.getId() + 1) + "/" +
getNumberOfSections()))
  );

  if (hasParameter("operation-comment")) {
    c1 += makeRow(
      makeColumn(d(localize("Description") + ": ") + v(getParameter("operation-
comment")))
    );
  }

  if (hasParameter("operation-strategy")) {
    var strategies = {
      drill: localize("Drilling"),
      face: localize("Facing"),
      path3d: localize("3D Path"),
      pocket2d: localize("Pocket 2D"),
      contour2d: localize("Contour 2D"),
```

```
    adaptive2d: localize("Adaptive 2D"),
    slot: localize("Slot"),
    circular: localize("Circular"),
    bore: localize("Bore"),
    thread: localize("Thread"),

    contour_new: localize("Contour"),
    contour: localize("Contour"),
    parallel_new: localize("Parallel"),
    parallel: localize("Parallel"),
    pocket_new: localize("Pocket"),
    pocket: localize("Pocket"),
    adaptive: localize("Adaptive"),
    horizontal_new: localize("Horizontal"),
    horizontal: localize("Horizontal"),
    flow: localize("Flow"),
    morph: localize("Morph"),
    pencil_new: localize("Pencil"),
    pencil: localize("Pencil"),
    project: localize("Project"),
    ramp: localize("Ramp"),
    radial_new: localize("Radial"),
    radial: localize("Radial"),
    scallop_new: localize("Scallop"),
    scallop: localize("Scallop"),
    morphed_spiral: localize("Morphed Spiral"),
    spiral_new: localize("Spiral"),
    spiral: localize("Spiral"),
    swarf5d: localize("Multi-Axis Swarf"),
    multiAxisContour: localize("Multi-Axis Contour")
  };
  var description = "";
  if (strategies[getParameter("operation-strategy")]) {
    description = strategies[getParameter("operation-strategy")];
  } else {
    description = localize("Unspecified");
  }
  c1 += makeRow(
    makeColumn(d(localize("Strategy") + ": ") + v(description))
  );
}
```

```
  var newWCS = !isFirstSection() && (currentSection.workOffset != getPreviousSection().workOffset);
  c1 += makeRow(
    makeColumn(d(localize("WCS") + ": ") + v(formatWCS(currentSection.workOffset) + (newWCS ? (" " + bold(localize("NEW!"))) : "")))
  );
  if (currentSection.isPatterned()) {
    var id = patternIds[currentSection.getPatternId()];
    c1 += makeRow(
      makeColumn(d(localize("Pattern Group") + ": ") + v(id))
    );
  }


 // var tolerance = cachedParameters["operation:tolerance"];
  var stockToLeave = cachedParameters["operation:stockToLeave"];
  var axialStockToLeave = cachedParameters["operation:verticalStockToLeave"];
  var maximumStepdown = cachedParameters["operation:maximumStepdown"];
  var maximumStepover = cachedParameters["operation:maximumStepover"] ? cachedParameters["operation:maximumStepover"] : cachedParameters["operation:stepover"];
  var optimalLoad = cachedParameters["operation:optimalLoad"];
  var loadDeviation = cachedParameters["operation:loadDeviation"];
/*
  if (tolerance != undefined) {
    c1 += makeRow(makeColumn(d(localize("Tolerance") + ": ") + v(spatialFormat.format(tolerance) + getUnitSymbolAsString())));
  }
*/
  if (stockToLeave != undefined) {
    if ((axialStockToLeave != undefined) && (stockToLeave != axialStockToLeave)) {
      c1 += makeRow(
        makeColumn(
          d(localize("Stock to Leave") + ": ") + v(spatialFormat.format(stockToLeave) + getUnitSymbolAsString()) + "/" + v(spatialFormat.format(axialStockToLeave) + getUnitSymbolAsString())
        )
      );
    } else {
      c1 += makeRow(makeColumn(d(localize("Stock to Leave") + ": ") + v(spatialFormat.format(stockToLeave) + getUnitSymbolAsString())));
    }
  }
/*
  if ((maximumStepdown != undefined) && (maximumStepdown > 0)) {
```

```
    c1 += makeRow(makeColumn(d(localize("Maximum stepdown") + ": ") + v(spatialFormat.format(maximumStepdown) + getUnitSymbolAsString())));
  }
*/
  if ((optimalLoad != undefined) && (optimalLoad > 0)) {
    c1 += makeRow(makeColumn(d(localize("Optimal load") + ": ") + v(spatialFormat.format(optimalLoad) + getUnitSymbolAsString())));
    if ((loadDeviation != undefined) && (loadDeviation > 0)) {
      c1 += makeRow(makeColumn(d(localize("Load deviation") + ": ") + v(spatialFormat.format(loadDeviation) + getUnitSymbolAsString())));
    }
  } else if ((maximumStepover != undefined) && (maximumStepover > 0)) {
    c1 += makeRow(makeColumn(d(localize("Maximum stepover") + ": ") + v(spatialFormat.format(maximumStepover) + getUnitSymbolAsString())));
  }


  var compensationType = hasParameter("operation:compensationType") ? getParameter("operation:compensationType") : "computer";
  if (compensationType != "computer") {
    var compensationDeltaRadius = hasParameter("operation:compensationDeltaRadius") ? getParameter("operation:compensationDeltaRadius") : 0;


    var compensation = hasParameter("operation:compensation") ? getParameter("operation:compensation") : "center";
    var COMPENSATIONS = {left: localize("left"), right: localize("right"), center: localize("center")};
    var compensationText = localize("unspecified");
    if (COMPENSATIONS[compensation]) {
      compensationText = COMPENSATIONS[compensation];
    }


    var DESCRIPTIONS = {computer: localize("computer"), control: localize("control"), wear: localize("wear"), inverseWear: localize("inverse wear")};
    var description = localize("unspecified");
    if (DESCRIPTIONS[compensationType]) {
      description = DESCRIPTIONS[compensationType];
    }
    c1 += makeRow(makeColumn(d(localize("Compensation") + ": ") + v(description + " (" + compensationText + ")")));
    c1 += makeRow(makeColumn(d(localize("Safe Tool Diameter") + ": ") + v("< " + spatialFormat.format(tool.diameter + 2 * compensationDeltaRadius) + getUnitSymbolAsString())));
  }
  c1 += "</table>";


  var c2 = "<table class=\"info\" cellspacing=\"0\">";
```

```
  c2 += makeRow(makeColumn(v(" "))); // move 1 row down
/*
  if (is3D()) {
    var zRange = currentSection.getGlobalZRange();
    c2 += makeRow(makeColumn(d(localize("Maximum Z") + ": ") + v(spatialFormat.format(zRange.getMaximum()) + getUnitSymbolAsString())));
    c2 += makeRow(makeColumn(d(localize("Minimum Z") + ": ") + v(spatialFormat.format(zRange.getMinimum()) + getUnitSymbolAsString())));
  }
*/
  var maximumFeed = currentSection.getMaximumFeedrate();
  var maximumSpindleSpeed = currentSection.getMaximumSpindleSpeed();
  var cuttingDistance = currentSection.getCuttingDistance();
  var rapidDistance = currentSection.getRapidDistance();
  var cycleTime = currentSection.getCycleTime();

  if (currentSection.getType() == TYPE_TURNING) {
    if (currentSection.getTool().getSpindleMode() == SPINDLE_CONSTANT_SURFACE_SPEED) {
      c2 += makeRow(makeColumn(d(localize("Surface Speed") + ": ") + v(rpmFormat.format(currentSection.getTool().surfaceSpeed * ((unit == MM) ? 1/1000.0 : 1/12.0)) + ((unit == MM) ? localize("m/min") : localize("ft/min")))));
    } else {
      c2 += makeRow(makeColumn(d(localize("Maximum Spindle Speed") + ": ") + v(rpmFormat.format(maximumSpindleSpeed) + localize("rpm"))));
    }
    if (currentSection.feedMode == FEED_PER_REVOLUTION) {
      if (hasParameter("operation:tool_feedCuttingRel")) {
        var feed = getParameter("operation:tool_feedCuttingRel");
        c2 += makeRow(makeColumn(d(localize("Feedrate per Rev") + ": ") + v(feedFormat.format(feed) + getFPRSymbolAsString())));
      }
    } else {
      c2 += makeRow(makeColumn(d(localize("Maximum Feedrate") + ": ") + v(feedFormat.format(maximumFeed) + getFeedSymbolAsString())));
    }
  } else {
    c2 += makeRow(makeColumn(d(localize("Maximum Spindle Speed") + ": ") + v(rpmFormat.format(maximumSpindleSpeed) + localize("rpm"))));
    c2 += makeRow(makeColumn(d(localize("Maximum Feedrate") + ": ") + v(feedFormat.format(maximumFeed) + getFeedSymbolAsString())));
  }
/*
  c2 += makeRow(makeColumn(d(localize("Cutting Distance") + ": ") + v(spatialFormat.format(cuttingDistance) + getUnitSymbolAsString())));
  if (properties.showRapidDistance) {
```

```
    c2    +=    makeRow(makeColumn(d(localize("Rapid    Distance")    +    ":    ")    +
v(spatialFormat.format(rapidDistance) + getUnitSymbolAsString())));
  }


  if (properties.rapidFeed > 0) {
    cycleTime += rapidDistance/properties.rapidFeed * 60;
  }
*/
  var additional = "";
  if ((getNumberOfSections() > 1) && properties.showPercentages) {
    if (totalCycleTime > 0) {
      additional          =          "<div          class=\"percentage\">("          +
percentageFormat.format(cycleTime/totalCycleTime) + "%)</div>";
    }
  }


  c2   +=   makeRow(makeColumn(d(localize("Estimated   Cycle   Time")   +   ":   ")   +
v(formatCycleTime(cycleTime) + " " + additional)));
  c2      +=      makeRow(makeColumn(d(localize("Coolant")      +      ":      ")      +
v(getCoolantDescription(tool.coolant))));


  c2 += "</table>";


  var c3 = "<table class=\"info\" cellspacing=\"0\">";
  c3 += makeRow(makeColumn(" "));
  c3 += makeRow(
    makeColumn(
      bold(localize("T") + toolFormat.format(tool.number)) + " " +
      localize("D") + toolFormat.format(tool.diameterOffset) + " " +
      localize("L") + toolFormat.format(tool.lengthOffset)
    )
  );


  c3      +=      makeRow(makeColumn(d(localize("Type")      +      ":      ")      +
v(getToolTypeName(tool.type))));
  c3      +=      makeRow(makeColumn(d(localize("Diameter")      +      ":      ")      +
v(spatialFormat.format(tool.diameter) + getUnitSymbolAsString())));
  if (tool.cornerRadius) {
    c3    +=    makeRow(makeColumn(d(localize("Corner    Radius")    +    ":    ")    +
v(spatialFormat.format(tool.cornerRadius) + getUnitSymbolAsString())));
  }
  if (tool.taperAngle) {
    if (tool.isDrill()) {
```

```
      c3    +=    makeRow(makeColumn(d(localize("Tip    Angle")    +    ":    ")    +
v(taperFormat.format(tool.taperAngle) + "&deg;")));
    } else {
      c3    +=    makeRow(makeColumn(d(localize("Taper    Angle")    +    ":    ")    +
v(taperFormat.format(tool.taperAngle) + "&deg;")));
    }
  }
  if (tool.isDrill() && (tool.threadPitch != 0)) {
    c3      +=      makeRow(makeColumn(d(localize("Pitch")      +      ":      ")      +
v(pitchFormat.format(tool.threadPitch)    +    getUnitSymbolAsString()    +    "/"    +
localize("turn"))));
  }
  c3      +=      makeRow(makeColumn(d(localize("Length")      +      ":      ")      +
v(spatialFormat.format(tool.bodyLength) + getUnitSymbolAsString())));
  c3 += makeRow(makeColumn(d(localize("Flutes") + ": ") + v(tool.numberOfFlutes)));
  if (tool.description) {
    c3 += makeRow(makeColumn(d(localize("Description") + ": ") + v(tool.description)));
  }
  if (tool.comment) {
    c3 += makeRow(makeColumn(d(localize("Comment") + ": ") + v(tool.comment)));
  }
  if (tool.holderDescription) {
    c3      +=      makeRow(makeColumn(d(localize("Holder")      +      ":      ")      +
v(tool.holderDescription)));
  }
  if (tool.aggregateId) {
    c3 += makeRow(makeColumn(d(localize("Aggregate") + ": ") + v(tool.aggregateId)));
  }
  if (tool.manualToolChange) {
    c3 += makeRow(makeColumn(d(bold(localize("Manual tool change")))));
  }
  c3 += "</table>";

  var c4 = "";
  if (toolRenderer && properties.showToolImage) {
    var id = useToolNumber ? tool.number : (currentSection.getId() + 1);
    var path = "tool" + id + ".png";
    var width = 2.5 * 100;
    var height = 2.5 * 133;
    try {
      if (!exportedTools[id]) {
        toolRenderer.exportAs(path, "image/png", tool, width, height);
        exportedTools[id] = true; // do not export twice
      }
```

```
      c4 = "<table class=\"info\" cellspacing=\"0\">" +
        makeRow("<td class=\"image\"><img width=\"100px\" src=\"" + path + "\"/></td>")
+
        "</table>";
    } catch(e) {
    }
  }

  write(
    "<tr class=\"info\">" +
      "<td valign=\"top\">" + c1 + "</td>" +
      "<td valign=\"top\">" + c2 + "</td>" +
      "<td valign=\"top\">" + c3 + "</td>" +
      (c4 ? "<td valign=\"top\" align=\"center\">" + c4 + "</td>" : "") +
    "</tr>"
  );

  if (properties.showPreviewImage) {
    var patternId = currentSection.getPatternId();
    var show = false;
    if (properties.forcePreview || !seenPatternIds[patternId]) {
      show = true;
      seenPatternIds[patternId] = true;
    }
    if (show && currentSection.hasParameter("autodeskcam:preview-name")) {
      var path = currentSection.getParameter("autodeskcam:preview-name");
      if
(FileSystem.isFile(FileSystem.getCombinedPath(FileSystem.getFolderPath(getOutputPath(
)), path))) {
        var r2 = "<table class=\"info\" cellspacing=\"0\">" +
          makeRow("<td class=\"preview\"><img src=\"" + path + "\"/></td>") +
          "</table>";
        write(
          "<tr class=\"info\">" +
          "<td colspan=\"4\" valign=\"top\" align=\"center\">" + r2 + "</td>" +
          "</tr>"
        );
      }
    }
  }

  if (properties.showNotes && hasParameter("notes")) {
    var notes = getParameter("notes");
```

```
    if (notes) {
      write(
        "<tr class=\"notes\"><td valign=\"top\" colspan=\"4\">" +
        d(localize("Notes")) + ": <pre>" + getParameter("notes") +
        "</pre></td></tr>"
      );
    }
  }

  if (!isLastSection()) {
    write("<tr
class=\"space\"><td> </td><td> </td><td> </td><td> </td></tr>");
  }
  writeln("");
  writeln("");

  cachedParameters = {};
}

function formatCycleTime(cycleTime) {
  cycleTime = cycleTime + 0.5; // round up
  var seconds = cycleTime % 60 | 0;
  var minutes = ((cycleTime - seconds)/60 | 0) % 60;
  var hours = (cycleTime - minutes * 60 - seconds)/(60 * 60) | 0;
  if (hours > 0) {
    return subst(localize("%1h:%2m:%3s"), hours, minutes, seconds);
  } else if (minutes > 0) {
    return subst(localize("%1m:%2s"), minutes, seconds);
  } else {
    return subst(localize("%1s"), seconds);
  }
}

function writeTotals() {
  var zRange;
  var maximumFeed = 0;
  var maximumSpindleSpeed = 0;
  var cuttingDistance = 0;
  var rapidDistance = 0;
  var cycleTime = 0;

  var numberOfSections = getNumberOfSections();
```

```
  var currentTool;
  for (var i = 0; i < numberOfSections; ++i) {
    var section = getSection(i);

    if (is3D()) {
      var _zRange = section.getGlobalZRange();
      if (zRange) {
        zRange.expandToRange(_zRange);
      } else {
        zRange = _zRange;
      }
    }

    maximumFeed = Math.max(maximumFeed, section.getMaximumFeedrate());
    maximumSpindleSpeed = Math.max(maximumSpindleSpeed,
section.getMaximumSpindleSpeed());
    cuttingDistance += section.getCuttingDistance();
    rapidDistance += section.getRapidDistance();
    cycleTime += section.getCycleTime();
    if (properties.toolChangeTime > 0) {
      var tool = section.getTool();
      if (currentTool != tool.number) {
        currentTool = tool.number;
        cycleTime += properties.toolChangeTime;
      }
    }
  }
  if (properties.rapidFeed > 0) {
    cycleTime += rapidDistance/properties.rapidFeed * 60;
  }
  totalCycleTime = cycleTime;

  writeln("<table class=\"sheet\" cellspacing=\"0\" align=\"center\">");
  write(makeRow("<th>" + localize("Total") + "</th>"));

  var c1 = "<table class=\"info\" cellspacing=\"0\">";
  var tools = getToolTable();
  c1 += makeRow(makeColumn(d(localize("Number Of Operations") + ": ") +
v(getNumberOfSections())));
  var text = "";
  for (var i = 0; i < tools.getNumberOfTools(); ++i) {
    var tool = tools.getTool(i);
```

```
    if (i > 0) {
      text += " ";
    }
    text += bold(localize("T") + toolFormat.format(tool.number));
  }
  c1 += makeRow(makeColumn(d(localize("Number Of Tools") + ": ") + v(tools.getNumberOfTools())));
  c1 += makeRow(makeColumn(d(localize("Tools") + ": ") + v(text)));
  if (zRange) {
  //       c1 += makeRow(makeColumn(d(localize("Maximum Z") + ": ") + v(spatialFormat.format(zRange.getMaximum()) + getUnitSymbolAsString())));
 //          c1 += makeRow(makeColumn(d(localize("Minimum Z") + ": ") + v(spatialFormat.format(zRange.getMinimum()) + getUnitSymbolAsString())));
  }
  c1 += makeRow(makeColumn(d(localize("Maximum Feedrate") + ": ") + v(feedFormat.format(maximumFeed) + getFeedSymbolAsString())));
  c1 += makeRow(makeColumn(d(localize("Maximum Spindle Speed") + ": ") + v(rpmFormat.format(maximumSpindleSpeed) + localize("rpm"))));
 //   c1 += makeRow(makeColumn(d(localize("Cutting Distance") + ": ") + v(spatialFormat.format(cuttingDistance) + getUnitSymbolAsString())));
  if (properties.showRapidDistance) {
 //        c1 += makeRow(makeColumn(d(localize("Rapid Distance") + ": ") + v(spatialFormat.format(rapidDistance) + getUnitSymbolAsString())));
  }
  c1 += makeRow(makeColumn(d(localize("Estimated Cycle Time") + ": ") + v(formatCycleTime(cycleTime))));
  c1 += "</table>";

  write(
    "<tr class=\"info\">" +
      "<td valign=\"top\">" + c1 + "</td>" +
    "</tr>"
  );
  write("</table>");
  writeln("");
  writeln("");
}

function onClose() {
  if (properties.showOperations) {
    writeln("</table>");
  }
  // footer
  if (properties.showFooter) {
```

```
    write("<br>");
    write("<div class=\"footer\">");
    var src = findFile("../graphics/logo.png");
    var dest = "logo.png";
    if (FileSystem.isFile(src)) {
      FileSystem.copyFile(src,
FileSystem.getCombinedPath(FileSystem.getFolderPath(getOutputPath()), dest));
      write("<img class=\"logo\" src=\"" + dest + "\"/>");
    }
    var now = new Date();
    var product = "Autodesk CAM";
    if (hasGlobalParameter("generated-by")) {
      product = getGlobalParameter("generated-by");
    }
    var productUrl = "http://cam.autodesk.com";
    if (product) {
      if ((product.indexOf("HSMWorks") == 0) || (product.indexOf("HSMXpress") == 0)) {
        productUrl = "http://www.hsmworks.com";
      }
    }
    write(localize("Generated by") + " <a href=\"" + productUrl + "\">" + product +
"</a>" + " " + now.toLocaleDateString() + " " + now.toLocaleTimeString());
    write("</div");
  }
  write("</body>");
  write("</html>");
}


function quote(text) {
  var result = "";
  for (var i = 0; i < text.length; ++i) {
    var ch = text.charAt(i);
    switch (ch) {
    case "\\":
    case "\"":
      result += "\\";
    }
    result += ch;
  }
  return "\"" + result + "\"";
}
```

```
function onTerminate() {
  // add this to print automatically - you could print to XPS and PDF writer
  /*
  var device = "Microsoft XPS Document Writer";
  if (device) {
    executeNoWait("rundll32.exe", "mshtml.dll,PrintHTML " + quote(getOutputPath()) +
quote(device), false, "");
  } else {
    executeNoWait("rundll32.exe", "mshtml.dll,PrintHTML " + quote(getOutputPath()),
false, "");
  }
  */
}
```

Příloha 2: Script DATA SHEET TOOL MANAGER

```
// Zdrojový kód – Form1.cs

namespace myPrintScreen
{
    public partial class Form1 : Form
    {
        private string saveFolder = @"C:\Dokumenty\printScreen\"; //složka do které
ukládám
       private int i = 0;

        // Import WinAPI funkcí zachytávající klávesy
        [DllImport("User32.dll")]
        private static extern short GetAsyncKeyState(int vKey);

       public Form1()
       {
              InitializeComponent();

           //zobrazí název složky , do které se ukládá
           label1.Text = saveFolder;
       }

        #region events buttons

        private void btnHide_Click(object sender, EventArgs e)
        {
            hideForm();
        }

        private void btnFolder_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start(saveFolder);
        }

        private void btnEnd_Click(object sender, EventArgs e)
        {
            Environment.Exit(0);
        }

        #endregion

        #region events

        private void Form1_Load(object sender, EventArgs e)
        {
            //testujeme, jestli to už nebylo spuštěno
            int CountProc = 0;
            foreach(Process p in Process.GetProcesses())
            {
                if (p.ProcessName == "myPrintScreen")
                    CountProc++;
            }
            //bylo spuštěno
            if (CountProc >1)
            {
                MessageBox.Show("Program byl spuštěn již dříve, použijte klávesu ESC
pro zobrazení ovládacího panelu");
                Environment.Exit(0);
            }


            // Spustí časovač
            timer1.Start();
```

```
        }


        /// <summary>
        ///  V rychlých intervalech kontroluje aktuálně stisknutou klávesu
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void timer1_Tick(object sender, EventArgs e)
        {
            string keyBuffer = "";
            // Snímá klávesy
            foreach (System.Int32 i in Enum.GetValues(typeof(Keys)))
            {
                if (GetAsyncKeyState(i) == -32767)
                {
                    keyBuffer = Enum.GetName(typeof(Keys), i);

                    // zpracování
                    switch (keyBuffer)
                    {
                        case "PrintScreen":
                            eventPrintScreen();
                            return;

                        case "Escape":
                            showForm();
                            return;
                    }
                }
            }
        }

       #endregion

       #region metods

        /// <summary>
        /// reakce na událost printScreen
        /// </summary>
        private void eventPrintScreen()
        {
            i++;
            string namePicture = string.Format("obr{0}.jpg", i);
            PrintScreen(namePicture);

        }

       /// <summary>
       /// uloží obsah obrazovky do JPG
       /// </summary>
       /// <param name="cesta"></param>
       void PrintScreen(string namePicture)
       {
              // Vytvoří bitmapu o stejných rozměrech jako primarni obrazovka
              Bitmap bmpScreenshot = new Bitmap(Screen.PrimaryScreen.Bounds.Width,

Screen.PrimaryScreen.Bounds.Height,

PixelFormat.Format32bppArgb);

              Graphics gfxScreenshot = Graphics.FromImage(bmpScreenshot);

              // Překopíruje obsah obrazovky do bitmapy
              gfxScreenshot.CopyFromScreen(Screen.PrimaryScreen.Bounds.X,
```

```
Screen.PrimaryScreen.Bounds.Y, 0, 0,

Screen.PrimaryScreen.Bounds.Size,

CopyPixelOperation.SourceCopy);

               // Uloží bitmapu do JPEG souboru
               bmpScreenshot.Save(saveFolder+namePicture);

       }

        /// <summary>
        /// skryje ovládací panel
        /// </summary>
        private void hideForm()
        {
             // Skryje aplikaci v pruhu úloh
            this.ShowInTaskbar = false;
            // Skryje okno aplikace
            this.Hide();
        }

        /// <summary>
        /// zobrazí ovládací panel
        /// </summary>
        private void showForm()
        {
            this.ShowInTaskbar = true;
            this.Show();
        }
       #endregion
    }
}

// Zdrojový kód – Form1.Designer.cs

namespace myPrintScreen
{
    partial class Form1
    {
       /// <summary>
       /// Required designer variable.
       /// </summary>
       private System.ComponentModel.IContainer components = null;

       /// <summary>
       /// Clean up any resources being used.
       /// </summary>
       /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
       protected override void Dispose(bool disposing)
       {
              if (disposing && (components != null))
              {
                     components.Dispose();
              }
              base.Dispose(disposing);
       }

       #region Windows Form Designer generated code

       /// <summary>
       /// Required method for Designer support - do not modify
       /// the contents of this method with the code editor.
```

```
        /// </summary>
        private void InitializeComponent()
        {
             this.components = new System.ComponentModel.Container();
             this.timer1 = new System.Windows.Forms.Timer(this.components);
             this.btnEnd = new System.Windows.Forms.Button();
             this.btnHide = new System.Windows.Forms.Button();
             this.btnFolder = new System.Windows.Forms.Button();
             this.label1 = new System.Windows.Forms.Label();
             this.button1 = new System.Windows.Forms.Button();
             this.button2 = new System.Windows.Forms.Button();
             this.button3 = new System.Windows.Forms.Button();
             this.checkedListBox1 = new System.Windows.Forms.CheckedListBox();
             this.button4 = new System.Windows.Forms.Button();
             this.listBox1 = new System.Windows.Forms.ListBox();
             this.textBox1 = new System.Windows.Forms.TextBox();
             this.textBox2 = new System.Windows.Forms.TextBox();
             this.textBox3 = new System.Windows.Forms.TextBox();
             this.SuspendLayout();
             //
             // timer1
             //
             this.timer1.Enabled = true;
             this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
             //
             // btnEnd
             //
             this.btnEnd.Location = new System.Drawing.Point(12, 187);
             this.btnEnd.Name = "btnEnd";
             this.btnEnd.Size = new System.Drawing.Size(132, 37);
             this.btnEnd.TabIndex = 0;
             this.btnEnd.Text = "Konec programu";
             this.btnEnd.UseVisualStyleBackColor = true;
             this.btnEnd.Click += new System.EventHandler(this.btnEnd_Click);
             //
             // btnHide
             //
             this.btnHide.Location = new System.Drawing.Point(12, 12);
             this.btnHide.Name = "btnHide";
             this.btnHide.Size = new System.Drawing.Size(129, 37);
             this.btnHide.TabIndex = 1;
             this.btnHide.Text = "Skrýt okno (znovu zobrazit klávesou ESC))";
             this.btnHide.UseVisualStyleBackColor = true;
             this.btnHide.Click += new System.EventHandler(this.btnHide_Click);
             //
             // btnFolder
             //
             this.btnFolder.Location = new System.Drawing.Point(9, 92);
             this.btnFolder.Name = "btnFolder";
             this.btnFolder.Size = new System.Drawing.Size(132, 34);
             this.btnFolder.TabIndex = 2;
             this.btnFolder.Text = "Otevřít složku";
             this.btnFolder.UseVisualStyleBackColor = true;
             this.btnFolder.Click += new System.EventHandler(this.btnFolder_Click);
             //
             // label1
             //
             this.label1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
             | System.Windows.Forms.AnchorStyles.Left)
             | System.Windows.Forms.AnchorStyles.Right)));
             this.label1.Location = new System.Drawing.Point(6, 56);
             this.label1.Name = "label1";
             this.label1.Size = new System.Drawing.Size(135, 33);
```

```
this.label1.TabIndex = 3;
this.label1.Text = "Cesta k .jpg";
this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
//
// button1
//
this.button1.Location = new System.Drawing.Point(150, 12);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(132, 37);
this.button1.TabIndex = 4;
this.button1.Text = "PrintScreen 1";
this.button1.UseVisualStyleBackColor = true;
//
// button2
//
this.button2.Location = new System.Drawing.Point(288, 12);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(132, 37);
this.button2.TabIndex = 5;
this.button2.Text = "PrintScreen 2";
this.button2.UseVisualStyleBackColor = true;
//
// button3
//
this.button3.Location = new System.Drawing.Point(426, 12);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(132, 37);
this.button3.TabIndex = 7;
this.button3.Text = "PrintScreen 3";
this.button3.UseVisualStyleBackColor = true;
//
// checkedListBox1
//
this.checkedListBox1.FormattingEnabled = true;
this.checkedListBox1.Items.AddRange(new object[] {
"PDF",
"XLS",
"HTTP"});
this.checkedListBox1.Location = new System.Drawing.Point(12, 132);
this.checkedListBox1.Name = "checkedListBox1";
this.checkedListBox1.Size = new System.Drawing.Size(129, 49);
this.checkedListBox1.TabIndex = 13;
//
// button4
//
this.button4.Location = new System.Drawing.Point(147, 187);
this.button4.Name = "button4";
this.button4.Size = new System.Drawing.Size(411, 37);
this.button4.TabIndex = 14;
this.button4.Text = "Generovat seřizovací list";
this.button4.UseVisualStyleBackColor = true;
//
// listBox1
//
this.listBox1.FormattingEnabled = true;
this.listBox1.Items.AddRange(new object[] {
"17:54:55\tPrintscreen 1 byl vytvořen",
"17:58:55\tPrintscreen 2 byl vytvořen",
"17:59:55\tPrintscreen 3 byl vytvořen",
"18:03:00\tByl vygenerován seřizovací list"});
this.listBox1.Location = new System.Drawing.Point(147, 86);
this.listBox1.Name = "listBox1";
this.listBox1.Size = new System.Drawing.Size(411, 95);
this.listBox1.TabIndex = 15;
//
```

```
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(150, 56);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(132, 20);
            this.textBox1.TabIndex = 16;
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(288, 56);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(132, 20);
            this.textBox2.TabIndex = 17;
            //
            // textBox3
            //
            this.textBox3.Location = new System.Drawing.Point(426, 56);
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(132, 20);
            this.textBox3.TabIndex = 18;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(564, 232);
            this.Controls.Add(this.textBox3);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.listBox1);
            this.Controls.Add(this.button4);
            this.Controls.Add(this.checkedListBox1);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.btnFolder);
            this.Controls.Add(this.btnHide);
            this.Controls.Add(this.btnEnd);
            this.Margin = new System.Windows.Forms.Padding(2);
            this.Name = "Form1";
            this.StartPosition = System.Windows.Forms.FormStartPosition.Manual;
            this.Text = "PrintScreen";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

         private System.Windows.Forms.Timer timer1;
         private System.Windows.Forms.Button btnEnd;
         private System.Windows.Forms.Button btnHide;
         private System.Windows.Forms.Button btnFolder;
         private System.Windows.Forms.Label label1;
         private System.Windows.Forms.Button button1;
         private System.Windows.Forms.Button button2;
         private System.Windows.Forms.Button button3;
         private System.Windows.Forms.CheckedListBox checkedListBox1;
         private System.Windows.Forms.Button button4;
         private System.Windows.Forms.ListBox listBox1;
         private System.Windows.Forms.TextBox textBox1;
         private System.Windows.Forms.TextBox textBox2;
         private System.Windows.Forms.TextBox textBox3;
```

```
    }
}


// Zdrojový kód - Program.cs

namespace myPrintScreen
{
    static class Program
    {
       /// <summary>
       /// The main entry point for the application.
       /// </summary>
       [STAThread]
       static void Main()
       {
              Application.EnableVisualStyles();
              Application.SetCompatibleTextRenderingDefault(false);
              Application.Run(new Form1());
       }
    }
}
```

Příloha 3: Script PROPOJENÍ v OnClose

```
function onClose() {
  if (properties.showOperations) {
    writeln("</table>");
  }

////////////////////////////////// Zobrazení příloh začátek
  writeln("<table class=\"sheet\" cellspacing=\"0\" align=\"center\">");

        var alignment = "center" ;
        var path="C:\\prenos\\screen1.bmp"
        if (FileSystem.isFile(path)) {
           write(makeRow("<th colspan=\"4\">Príloha1</th>"));
          var src = getImageAsImgSrc(path);
          write("<td class=\"model\" align=\"" + alignment + "\"><img src=\""
+ src + "\"/></td>");
         }

        var path="C:\\prenos\\screen2.bmp"
        if (FileSystem.isFile(path)) {
           write(makeRow("<th colspan=\"4\">Príloha2</th>"));
          var src = getImageAsImgSrc(path);
          write("<td class=\"model\" align=\"" + alignment + "\"><img src=\""
+ src + "\"/></td>");
         }

        var path="C:\\prenos\\screen3.bmp"
        if (FileSystem.isFile(path)) {
           write(makeRow("<th colspan=\"4\">Príloha3</th>"));
          var src = getImageAsImgSrc(path);
          write("<td class=\"model\" align=\"" + alignment + "\"><img src=\""
+ src + "\"/></td>");
         }

  writeln("</table>");
  writeln("");
////////////////////////////////////// Zobrazení příloh konec
```

```
  // footer
  if (properties.showFooter) {
    write("<br>");
    write("<div class=\"footer\">");
    var src = findFile("../graphics/logo.png");
    var dest = "logo.png";
    if (FileSystem.isFile(src)) {
      FileSystem.copyFile(src,
FileSystem.getCombinedPath(FileSystem.getFolderPath(getOutputPath()), dest));
      write("<img class=\"logo\" src=\"" + dest + "\"/>");
    }
    var now = new Date();
    var product = "Autodesk CAM";
    if (hasGlobalParameter("generated-by")) {
      product = getGlobalParameter("generated-by");
    }
    var productUrl = "http://cam.autodesk.com";
    if (product) {
      if ((product.indexOf("HSMWorks") == 0) || (product.indexOf("HSMXpress")
== 0)) {
        productUrl = "http://www.hsmworks.com";
      }
    }
    write(localize("Generated by") + " <a href=\"" + productUrl + "\">" +
product + "</a>" + " " + now.toLocaleDateString() + " " +
now.toLocaleTimeString());
    write("</div");
  }
  write("</body>");
  write("</html>");
}

function quote(text) {
  var result = "";
  for (var i = 0; i < text.length; ++i) {
    var ch = text.charAt(i);
    switch (ch) {
    case "\\":
    case "\"":
      result += "\\";
    }
    result += ch;
  }
  return "\"" + result + "\"";
}

function onTerminate() {
  // add this to print automatically - you could print to XPS and PDF writer
  /*
  var device = "Microsoft XPS Document Writer";
  if (device) {
    executeNoWait("rundll32.exe", "mshtml.dll,PrintHTML " +
quote(getOutputPath()) + quote(device), false, "");
  } else {
    executeNoWait("rundll32.exe", "mshtml.dll,PrintHTML " +
quote(getOutputPath()), false, "");
  }
  */
}
```