A .NET application searching for data in a log file of the KUKA industrial welding robot

Igor Košťál*

* University of Economics in Bratislava, Faculty of Economic Informatics, Dolnozemská cesta 1, 852 35 Bratislava, Slovakia, e-mail: *igor.kostal@euba.sk*

Abstract—**In a robotic workstation designed and created from KUKA industrial robots, manipulators and its control, each of these robots is equipped with some tool, for example, welding pliers. This tool is mounted on the mounting flange of a robot. During tool calibration, the robot operator assigns the working point to this tool. This point is called the TCP (Tool Center Point). All motions of the TCP are controlled by a robot program. By this is addressed through which points the TCP passes during its motion. Each particular KUKA industrial robot of an assembled robotic workstation is tested by the operator in the test operation. The precision of the TCP motion is measured and it is represented by points through which the TCP passes. During testing of a certain activity of a robot it happens many times that the operator finds that the TCP of a robot tool does not perform motions along the prescribed path with a sufficient precision. Therefore he must add other points to this path to refine it. It happens also often that during testing of a certain activity of a robot the operator must change the coordinates of some points, sometimes several times, to refine or to adjust the motion path of the TCP of a robot tool. These additional and modified points identify problematic spots of a motion path of the TCP. The operator performs all additions of new points and changes of the coordinates of existing points of the TCP path through the KCP (KUKA Control Panel) teach pendant in a robot program in the KUKA Robot Language (KRL) and the KUKA System Software (KSS) records them into the appropriate files to the hard drive of the robot control PC. In addition, all the operator actions on the KCP are automatically logged by KSS and he can generate the robot log file *Logbuch.txt* by KSS. All added and modified path points of the TCP in the robot program during the test operation of the robot are recorded into this log file, too. These points provide important information about the TCP path sections for the operator, with which were problems during testing of a certain activity of the robot. During retesting of this activity of the robot it is therefore necessary to focus specially on these sections of the path, i.e. on the points forming these sections of the path. Thus, the operator quickly needs find out which points he added to the TCP path and also coordinates of points of this path which were repeatedly modified in the robot control program. We have created the .NET search application that is able to connect to the control PC of the KUKA industrial welding robot and find on its disk the log file *Logbuch.txt*. Then, in this file, according to the user instructions it searches for and summarizes relevant new founded and repeatedly modified the TCP path points in the robot program during test operation of selected activity of a robot. The paper deals with data files of the robot program in which are stored coordinates of the path points of the TCP robot tool, and with outputs of our .NET search application searching this log file of the KUKA industrial welding robot.**

Keywords-robot; Tool Center Point; KRL program; motion programming; log file; testing and retesting robot

I. Introduction

Creating a robotic workstation built from KUKA industrial robots, manipulators and their control consists of the following steps:

- *the design of the robotic workstation* – a technical specification of particular robots and manipulators of the workstation, their arrangement, the design of the paths of robots tools motions, etc.. This activity is carried out in specialized software, for example, in Robcad.
- *programming particular robots and manipulators of the robotic workstation* in specialized software, for example, in Orange edit
- *the design and programming a control (higher-level controllers, e.g. PLC) of the whole robotic workstation* in specialized software, for example, in SIMATIC STEP 7
- *assembling, putting into operation and testing the robotic workstation in a developer company* – a professional installation of robots, manipulators and a control of the robotic workstation, inserting the created control software into control computers of robots and manipulators and into a control device of robotic workstation (a higher-level controller, e.g. PLC), a start-up of the workstation and its testing.
- *assembling, putting into operation and testing the robotic workstation at a customer* - there are carried out the same activities as in the developer company.
- *start-up of a serial production in the robotic workstation at a customer* - if all required tests of the robotic workstation at a customer have been carried out with satisfactory results, the robotic workstation is submitted to the customer.

Testing of particular robots of a robotic workstation is a very important and necessary part of two stages of creating such workstation. It affects the work quality of particular robots in a serial production in robotic workstation fundamentally. We have created the .NET search application that makes testing and retesting of certain activities of the KUKA industrial welding robot more effectively and makes this process faster. This robot is the part of robotic workstation welding a cooling system radiator bracket of a passenger car. The paper deals with the functioning of our .NET search application and outputs that this application provides to the robot operator during testing and retesting of certain activities of this robot.

II. Programming a KUKA Industrial Robot

All activities of each particular KUKA industrial robot of a robotic workstation are controlled by the control program of this robot. Control programs of KUKA industrial robots are created in the KUKA Robot Language (KRL). Its syntax allows developers motion programming, a program execution control, programming inputs/outputs, programming subprograms and functions, etc.

We can create two kinds of KRL programs [1]:

- a motion program
- a SUB program, that can perform operator control or monitoring tasks, for example, monitoring of a robot safety equipment or monitoring of a robot cooling circuit. However, some KRL instructions can be used in a SUB program. These programs are always files with the extension *.SUB.

The motion program runs in the robot interpreter and a SUB program runs in the Submit interpreter. Both interpreters run in parallel on the robot control PC. [1]

A KRL motion program generally consists of an SRC file and a DAT file of the same name. An SRC file contains the program code and a DAT file contains permanent data and point coordinates. The DAT file is also called a data list. The SRC file and associated DAT file together are called a module. [1]

Our .NET search application works with the SRC and DAT files of robot motion programs, therefore, we describe the motion statements that are used in these SRC files briefly.

The following robot motions types can be programmed: *Point-to-point motions (PTP)*, *Linear motions (LIN)*, *Circular motions (CIRC)*, *Spline motions*. [1]

LIN, CIRC and spline motions are also known as CP (“Continuous Path”) motions. [1]

The start point of a motion is always the end point of the previous motion. [1]

In our robotic workstation each of robots is equipped with some tool, for example, welding pliers. This tool is mounted on the mounting flange of a robot. During tool calibration, the user assigns a Cartesian coordinate system (TOOL coordinate system) to this tool [1]. The TOOL coordinate system has its origin at a user-defined point [1]. This is called the TCP (Tool Center Point). The TCP is generally situated at the working point of the tool [1]. In motion programming we program the motion of the TCP of robot tool.

Motion type PTP [1]

The robot guides the TCP along the fastest path to the end point. The fastest path is generally not the shortest path and is thus not a straight line. As the motions of the robot axes are rotational, curved paths can be executed faster than straight paths.

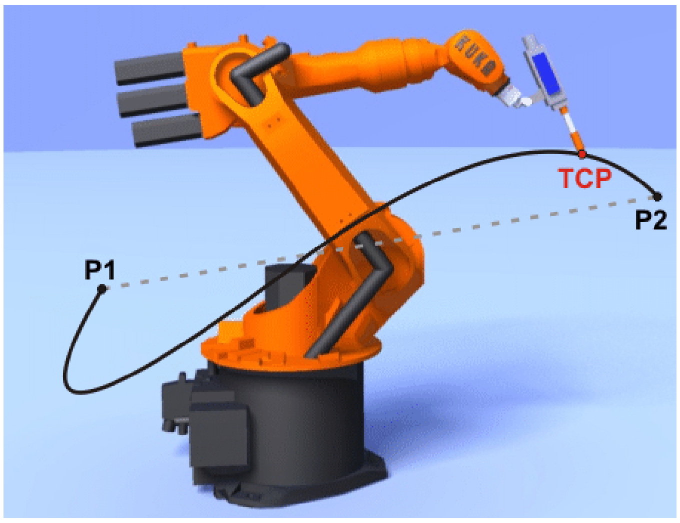

Figure 1. PTP motion [1]

Motion type LIN [1]

The robot guides the TCP at a defined velocity along a straight path to the end point.

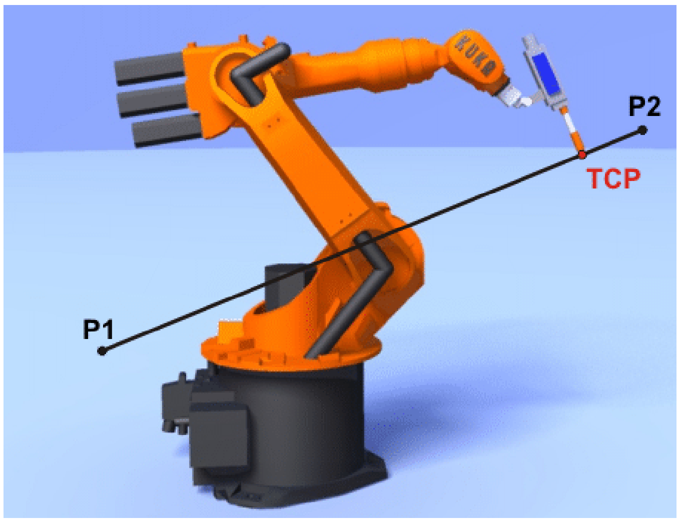

Figure 2. LIN motion [1]

PTP statement [1]

executes a point-to-point motion to the end point. The coordinates of the end point are absolute.

Syntax

PTP *End_Point* <C_PTP <*Approximate_Positioning*>>

End_Point - type: POS, E6POS, AXIS, E6AXIS, FRAME. The end point can be specified in Cartesian or axis-specific coordinates. Cartesian coordinates refer to the BASE coordinate system.

C_PTP - causes the end point to be approximated. The specification C_PTP is sufficient for PTP-PTP approximate positioning. In the case of PTP-CP approximation, i.e. if the approximated PTP block is followed by a LIN or CIRC block, *Approximate_Positioning* must also be specified.

Approximate_Positioning - only for PTP-CP approximate positioning. This parameter defines the earliest point at which the approximate positioning can begin.

The PTP statement example

End point is specified in Cartesian coordinates. X, Y, Z define the position and A, B, C define orientation of the TCP. S (Status) and T (Turn) define axis positions unambiguously. Used units: X, Y, Z [mm]; A, B, C [deg]; S, T [bin].

PTP {X 12.3, Y 100.0, Z 50, A 9.2, B 50, C 0,S 'B010',T 'B1010'}

LIN statement [1]

executes a linear motion to the end point. The coordinates of the end point are absolute.

Syntax

LIN *End_Point* <*Approximate_Positioning*>

End_Point - type: POS, E6POS, FRAME. The Status and Turn specifications for an end point of type POS or E6POS are ignored in the case of LIN motions. The coordinates refer to the BASE coordinate system.

Approximate_Positioning - This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin.

The LIN statement example

End point with two components. For the rest of the components, the controller takes the values of the previous position.

LIN {Z 500, X 123.6}

III. A .NET Application Searching a Log File of the KUKA Industrial Welding Robot

As we mentioned above, testing particular robots of a robotic workstation is a very important and necessary part of two stages of creating such workstation because it fundamentally affects the particular robots work quality in a serial production in this workstation.

During the testing of certain activities of each KUKA industrial welding robot of our robotic workstation the precision of the TCP these robots motion is measured. It is represented by points through which the TCP passes. It happens many times that during testing of a certain activity of a robot the operator finds that the TCP of a robot tool does not perform motions along the prescribed path with a sufficient precision. Therefore he must add other points to this path to refine it. It happens also often that during testing of a certain activity of a robot the operator must change the coordinates of some points, sometimes several times, to refine or to adjust the motion path of the TCP of a robot tool. These additional and modified points identify problematic spots of a motion path of the TCP. The operator performs all additions of new points and changes of the coordinates of existing points of the TCP path through the KCP (KUKA Control Panel) teach pendant in a robot control program in KRL. All changes in the robot control program are stored into its SRC and DAT files placed on the hard drive of the robot control PC. In addition, all the operator actions on the KCP are automatically logged by the KUKA System Software (KSS) and he can generate the robot log file *Logbuch.txt* by KSS. All added and modified path points of the TCP in the robot control program during the test operation of the robot are recorded into this log file, too. These points provide important information about the TCP path sections for the operator, with which were problems during testing of a certain activity of the robot. During retesting of this activity of the robot it is therefore necessary to focus specially on these sections of the path, i.e. on the points forming these sections of the path. Thus, the operator quickly needs to find out which points he added to the TCP path and also coordinates of points of this path which were repeatedly modified in the robot control program.

We have created the .NET search application that can find such information in the *Logbuch.txt* file effectively and provide them to the user.

Our .NET search application was developed in the C# language in the development environment Microsoft Visual Studio 2013 for the Microsoft .NET Framework version 4. Therefore this framework must be installed in the compatible operating system, for example Microsoft Windows 7 Home Premium, 64 bit, of the computer on which our application runs. This .NET application searches for required data of three KRL programs with names *servisna_pozicia* (*service_position*), *frezovanie_capic* (*milling_caps*) and *vymena_capic* (*replacement_caps*) in the log file *Logbuch.txt.* These KRL programs are part of the KUKA industrial welding robot control program. Using these programs the robot carries out activities related to the programs names. It means, if the cover caps of robot welding pliers spikes are worn, the TCP of the robot moves to the specified position, where these cover caps are milled (the program *frezovanie_capic* (*milling_caps*)). If these cover caps are worn more than the permissible value, they are exchanged for new ones (the program *vymena_capic* (*replacement_caps*)) in the TCP service position (the program *servisna_pozicia* (*service_position*)).

Immediately after the start-up our .NET search application on the user's computer it attempts to connect through the Intranet to the robot control PC and searches for the directory with the *Logbuch.txt* file and the directory with DAT and SRC files of above mentioned programs on the robot control PC hard drive. If the .NET search application does not find these directories for some reasons, then it calls the user to search for them through a dialog box. When the .NET search application is connected to the correct directories on the robot control PC hard drive, it will create items of both its combo boxes dynamically. The left combo box will contain items with SRC and DAT files names from the found directory. The .NET search application is ready to full use now.

For each of the three KRL programs *frezovanie_capic* (*milling_caps*), *servisna_pozicia* (*service_position*) and *vymena_capic* (*replacement_caps*) our .NET search application provides the following outputs to the user:

- *The basic statistics* about **new-founded points** in the DAT file of the relevant program, which the .NET search application found in the *Logbuch.txt* file: the total number of all logs in the *Logbuch.txt* file, the number of logs *New point founded* in the DAT file of the relevant program, which were found in the *Logbuch.txt* file, the list of all points and new-founded points in the DAT file of the relevant program. The .NET search application displays these statistics in a text box.

  It displays ***all found complete logs*** about **new-founded points** in the DAT file of the relevant program as items in a checked list box control.

  Besides that the .NET search application displays its search results in the form of basic statistics and complete found requested logs in its two controls, it also creates the disk file with the name, for example, *NEW_FOUNDED_points_in_ VYMENA_CAPIC_DAT_20140921_153338.DAT* (*NEW_FOUNDED_points_in_REPLACEMENT_CAPS_ DAT_20140921_ 153338.DAT*) containing the date and time when the file was created (2014-09-21 15:33:38). This file will contain the complete search results.

```
The results of searching for NEW-FOUNDED points in the VYMENA_CAPIC.DAT
                       file (from 2014-09-21 15:33:38Z)

The TOTAL number of ALL logs in the 'Logbuch.txt' file                    : 5281
The number of logs 'New point founded' found in VYMENA_CAPIC.DAT : 2
The list of ALL points :  XP1, XP2, XP3, XP4, XP5, XP6, XP7, XP8, XP9, XP10,
              XP11, XP12, XP13, XP14, XP15, XP16, XP17, XP18, XP19, XP20
The list of NEW-FOUNDED points :  XP12, XP13
------------------------------------------------------------------------------------------------------------

   The list of ALL full logs 'New point founded' found in VYMENA_CAPIC.DAT:

Log 327 (User action, Error)
2014-01-15 06:46:27'789
New point founded
Point name: /R1/VYMENA_CAPIC.DAT/XP13
Coordinates:  {E6POS: X -931.7312, Y 871.5461, Z 1381.744, A 91.86456,
              B 3.368453, C -132.3256, S 2, T 43, E1 -39.99435, E2 0.0, E3 0.0,
              E4 0.0, E5 0.0, E6 0.0}
Module: Techhandler   MsgNo: 1

Log 305 (User action, Error)
2014-01-15 06:44:21'872
New point founded
Point name: /R1/VYMENA_CAPIC.DAT/XP12
Coordinates:  {E6POS: X -931.7707, Y 871.5453, Z 1381.755, A 91.86541,
              B 3.367920, C -132.3256, S 2, T 43, E1 -39.99450, E2 0.0, E3 0.0,
              E4 0.0, E5 0.0, E6 0.0}
Module: Techhandler   MsgNo: 1
```

Figure 3. The file *NEW_FOUNDED_points_in_VYMENA_CAPIC_DAT_ 20140921_153338.DAT* with search results (the .NET search application displays the same outputs in its two controls)

- *The basic statistics* about **modified points** in the DAT file of the relevant program, which the .NET search application found in the *Logbuch.txt* file: the total number of all logs in the *Logbuch.txt* file, the number of logs *Point modified* in the DAT file of the relevant program, that were found in the *Logbuch.txt* file, the list of all points, modified and unchanged points in the DAT file of the relevant program and **the list of points in this DAT files that were modified more than one times with specified number of modifications**. The .NET search application displays these statistics in a text box.

  It displays ***all found complete logs*** about **modified points** in the DAT file of the relevant program as items in a check list box control with the **specified order of change for multiple modified points**. The operator can use this control for marking points, mainly problem points with the repeatedly coordinates modified, which he already tested during retesting the precision of the motion of the robot TCP. The items with complete logs about modified points checked by the operator using a check mark change the text color to red at the same time.

  Besides that the .NET search application displays its search results in the form of basic statistics and complete found requested logs in its two controls, it also creates the disk file with the name, for example, *MODIFIED_points_ in_VYMENA_CAPIC_DAT_20140921_183042.DAT* (*MODIFIED_points_in_REPLACEMENT_CAPS_DAT_ 20140921_183042.DAT*) containing the date and time when the file was created (2014-09-21 18:30:42). This file will contain the complete search results.

```
 The results of searching for MODIFIED points in the VYMENA_CAPIC.DAT file
                          (from 2014-09-21 18:30:42Z)

The TOTAL number of ALL logs in the 'Logbuch.txt' file                  : 5281
The number of logs 'Point modified ' found in VYMENA_CAPIC.DAT    : 14

The list of ALL points:  XP1, XP2, XP3, XP4, XP5, XP6, XP7, XP8, XP9, XP10,
              XP11, XP12, XP13, XP14, XP15, XP16, XP17, XP18, XP19, XP20
The modified points :  XP2, XP3, XP4, XP5, XP6, XP12, XP13 (The
                    UNCHANGED points: XP1, XP7, XP8, XP9, XP10, XP11,
                    XP14, XP15, XP16, XP17, XP18, XP19, XP20)
The points modified many times:
XP2 : 2 times     XP3 : 2 times     XP4 : 3 times     XP5 : 2 times     XP6 : 3 times
------------------------------------------------------------------------------------------------------------

        The list of ALL full logs 'Point modified' found in VYMENA_CAPIC.DAT:

Log 467 (User action, Error)
2014-01-15 07:14:20'328
Point modified
Point name: /R1/VYMENA_CAPIC.DAT/XP13
Old coordinates:   {E6POS: X -947.56813, Y 967.5863, Z 1392.765, A 99.96456,
                   B 3.339458, C -162.4836, S 2, T 42, E1 -47.99642, E2 0.0,
                   E3 0.0, E4 0.0, E5 0.0, E6 0.0}
New coordinates:  {E6POS: X -1009.787, Y 993.2896, Z 1441.781, A 98.73503,
                   B 9.315615E-01, C -153.0429, S 2, T 42, E1 -47.99632, E2 0.0,
                   E3 0.0, E4 0.0, E5 0.0, E6 0.0}
Module: Techhandler   MsgNo: 2

Log 445 (User action, Error)
2014-01-15 07:12:19'851
Point modified [The 2nd and FINAL modification]
Point name: /R1/VYMENA_CAPIC.DAT/XP5
Old coordinates:   {E6POS: X -541.8906, Y 671.3859, Z 2283.245, A 185.0567,
                   B 19.56994, C 213.2984, S 3, T 35, E1 -44.89536, E2 0.0,
                   E3 0.0, E4 0.0, E5 0.0, E6 0.0}
New coordinates:  {E6POS: X -572.8052, Y 632.2319, Z 2207.520, A 186.8900,
                   B 18.82320, C 218.0009, S 3, T 35, E1 -44.89506, E2 0.0,
                   E3 0.0, E4 0.0, E5 0.0, E6 0.0}
Module: Techhandler   MsgNo: 2

Log 443 (User action, Error)
2014-01-15 07:12:15'260
Point modified [The 3rd and FINAL modification]
Point name: /R1/VYMENA_CAPIC.DAT/XP6
Old coordinates:   {E6POS: X -541.8907, Y 671.3861, Z 2283.245, A 185.0567,
                   B 19.56993, C 213.2984, S 3, T 35, E1 -44.89531, E2 0.0,
                   E3 0.0, E4 0.0, E5 0.0, E6 0.0}
New coordinates:  {E6POS: X -572.8051, Y 632.2320, Z 2207.520, A 186.8900,
                   B 18.82318, C 218.0009, S 3, T 35, E1 -44.89506, E2 0.0,
                   E3 0.0, E4 0.0, E5 0.0, E6 0.0}
Module: Techhandler   MsgNo: 2
```

Figure 4. The part of the file *MODIFIED_points_in_ VYMENA_CAPIC_DAT_20140921_183042.DAT* with search results (the .NET search application displays the same outputs in its two controls)

- *The basic statistics* about **statements added** into the SRC file of the relevant program: the number of statements added into this SRC file, the number and names of new-founded points in the DAT file of the relevant program for which statements were added into this SRC file. The .NET search application found statements added into the SRC file of the relevant program according to the names of new-founded points in the DAT file of the relevant program, which logs were found by this application in the *Logbuch.txt* file before this search. The .NET search application displays these statistics in a text box.

  It displays ***all found complete statements***, which were **added into the SRC file** of the relevant program, as items in a checked list box control.

  Besides that the .NET search application displays its search results in the form of basic statistics and complete found statements added into the relevant SRC file in its two controls, it also creates the disk file with the name, for example, *statements_ADDED_into_FREZOVANIE_CAPIC_SRC_20140921_183252.DAT* (*statements_ADDED_into_MILLING_CAPS_SRC_20140921_183252.DAT*) containing the date and time when the file was created (2014-09-21 18:32:52). This file will contain the complete search results.

```
           The results of searching for statements ADDED into the
           FREZOVANIE_CAPIC.SRC file (from 2014-09-21 18:32:52Z)

The number of statements ADDED into the FREZOVANIE_CAPIC.SRC file : 11

The statements were ADDED into the FREZOVANIE_CAPIC.SRC file for
11 NEW-FOUNDED points in FREZOVANIE_CAPIC.DAT  : XP1, XP2, XP3, XP4,
                        XP5, XP6, XP7, XP8, XP9, XSG0000003, XSG0000032
-----------------------------------------------------------------------------------

    The list of ALL statements ADDED into the FREZOVANIE_CAPIC.SRC file:

;FOLD PTP P9 CONT Vel=100 % PDAT9 Tool[1]:SpotGun Base[0];%{PE}%R
7.2.24,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P9, 3:C_DIS, 5:100,
7:PDAT9
$BWDSTART=FALSE
PDAT_ACT=PPDAT9
FDAT_ACT=FP9
BAS(#PTP_PARAMS,100)
PTP XP9 C_DIS
;ENDFOLD

;FOLD PTP P8 CONT Vel=100 % PDAT8 Tool[1]:SpotGun Base[0];%{PE}%R
7.2.24,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P8, 3:C_DIS, 5:100,
7:PDAT8
$BWDSTART=FALSE
PDAT_ACT=PPDAT8
FDAT_ACT=FP8
BAS(#PTP_PARAMS,100)
PTP XP8 C_DIS
;ENDFOLD

;FOLD PTP P7 CONT Vel=100 % PDAT7 Tool[1]:SpotGun Base[0];%{PE}%R
7.2.24,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P7, 3:C_DIS, 5:100,
7:PDAT7
$BWDSTART=FALSE
PDAT_ACT=PPDAT7
FDAT_ACT=FP7
BAS(#PTP_PARAMS,100)
PTP XP7 C_DIS
;ENDFOLD
```

Figure 5. The part of the file *statements_ADDED_into_FREZOVANIE_CAPIC_SRC_20140921_183252.DAT* with search results (the .NET search application displays the same outputs in its two controls)

All DAT files created by the .NET search application that contain complete results of relevant searches, contain in the name and in its content a time stamp, which allows us to archive changes in SRC and DAT files of the relevant program in a temporal chronology, as they were carried out. It also allows for every testing to determine which points were newly founded or changed in the DAT file of the relevant program or which statements have been added to the SRC file of the relevant program during the given testing. All these information are very useful for the programmer and operator of the robot.

In addition, when we restart searches for the same requested data in the unchanged file *Logbuch.txt*, then the .NET search application creates a new file with the current date and time in the name and contents of the file, but with the same search results as the previous file of such search. Then this application deletes the previous file. By this the .NET search application prevents the creation of the DAT files with the same search results.

Very valuable output of the .NET search application for the robot operator is a list of modified and unchanged points in the DAT file of the relevant program and especially **the list of points in this DAT file that were modified more than one time with specified number of changes of each point**. In this last list, the operator has immediately available the selected list of points, with which he had problems during last testing, because he modified their coordinates several times; he refined their position many times. During retesting a precision of a given robot move the operator focuses just on these points and he does not need to deal with all modified points, which coordinates he modified during previous testing. The .NET search application also displays in check list box the old and new coordinates of all modified points, including several times modified points. It is also very valuable information because the operator can immediately see the development of changes of coordinates of modified and multiple modified points along with the specified ordinal number of a change.

To search all these valuable information manually in 5281 logs stored in the *Logbuch.txt* file is very ineffective and can also lead to oversight of some searched logs. In addition, during the repeated multiple testing of a robot it would be also tiring and time consuming work, which could slow down testing itself.

Our .NET search application is able to provide to the robot operator all above mentioned outputs quickly and repeatedly after each test of the robot and in addition, it stores all its search results into disk files with a time stamp, what is a great, above-described advantage, too.

IV. Conclusion

The most valuable search results of required logs in the *Logbuch.txt file* that our .NET search application provides to the user are:

- the list of modified points in the DAT file of the relevant program,

- the list of multiple modified points in the DAT file of the relevant program with the specified number of changes and with the specified ordinal number of a change,
- old and new coordinates of the points from these two lists.

The first two mentioned lists give the operator clear information about the points on which he has to focus during retesting the precision of the TCP robot motion.

The results of all searches stored in the corresponding disk files make it easier for the operator to archive all changes done in the SRC and DAT files of relevant programs during repeated tests of the precision of the TCP robot motion, which were part of repeated testing of robots particular activities. Thus, our .NET search application can be a very effective software support for the operator during testing and retesting of robots particular activities.

References

[1] KUKA Roboter GmbH, KUKA System Software, KUKA System Software 5.5, Operating and Programming Instructions for System Integrators. KUKA Roboter GmbH, Augsburg, Germany, 2010. (references)

[2] KUKA Roboter GmbH, KUKA System Technology, KUKA.RobotSensorInterface 2.3 for KUKA System Software 5.4, 5.5, 5.6, 7.0. KUKA Roboter GmbH, Augsburg, Germany, 2009.

[3] KUKA Roboter GmbH, Controller, KR C4; KR C4 CK, Specification. KUKA Roboter GmbH, Augsburg, Germany, 2013.

[4] Microsoft Corporation 2014, “MSDN Library on-line,” http://msdn.microsoft.com.