An Integrated CAD/CAM System for Robotized Arc Welding

János Nacsa and George L. Kovács

CIM Research Laboratory
Computer and Automation Research Institute of Hungarian Academy of Sciences
P.O.B. 63, Budapest
HUNGARY
E-mails: nacsa@sztaki.hu, h673kov@ella.hu

Abstract: In the frame of the COPERNICUS EC-Program a CAD-based Programming System for arc welding robots was developed in an international project with German and Hungarian participants. The basic goal of the system is to assist robotized arc welding in Small and Medium Size Enterprises (SME) mostly for small and medium lot sizes. The product of the development is such an interactive off-line robot programming system that gives assistance to easily design and then perform arc welding tasks. Commercially available off-line robot programming systems are either too expensive or unable to provide all required functionalities. PROARC is implemented on the world-leading PC-based CAD system, AutoCAD, by using its new object-oriented interface ARX. However beside these platform related attributes of the system, PROARC provides a number of efficient functionalities for the seam programming. In order to allow the system to be integrated into a CIM environment, standardised OSI (Open System Interconnection) interfaces have been chosen. The paper gives a more detailed view about the defined STEP interface and the implemented OSI based network connection.

Keywords: CAD/CAM, arc welding, integration, MAP, MMS

1. Introduction

A large number of industrial robots has been established in the field of arc welding over the last twenty years. The reliability of robots, their use in more than two shifts and the reproducible welding quality are arguments for the use of this technology.

However, the application of arc welding robots is mostly limited to the production of big lot sizes. The classical way of programming a robot is still the teach-in method in which the robot has to be led to the various points, which will afterwards be connected by a manually generated program. This programming method is highly insufficient and hardly economical for small or medium lot sizes. Therefore off-line programming systems have been developed to decouple the process of program generation of the actual robot in the workcell.

Many different programming systems are available on the market which meet the different requirements of the users. At the beginning of the PROARC project more than ten systems were analysed [PROARC, 1994]. Here are some samples.

The Windows based WORKSPACE system from Robot Simulations [Workspace, 1996] is basically used for the simulation of robot applications. Without focusing on specific applications and providing tailored functionalities, this software especially concentrates on the calibration of the real and the modelled workcell. A special tool ROBOTRAK allows, together with the use of a measurement system, the update of the modelled cell and of its components with a high accuracy. This is especially required for applications, in which no on-line sensing is or can be used. However, in the case of arc welding the arc sensor during and the tactile sensor before the welding process guarantee an appropriate welding of the seams.

GRASP [BYG, 1994 ; 1995] is a 3D graphical simulation tool with off-line programming features including a welding assist menu. It has a well defined internal robot motion ASCII language called GRDATA. CAD interfaces and robot language postprocessors are available. Collision detection, kinematics modelling, automatic program generation, are also features of the system. This high end system is used in many applications.

AnySIM [ifp, 1994] is a general simulation tool which has a basic module and special extensions for simulating NC machines, human resources, etc. It has also robot simulation capacities, but no arc welding features are built in. AnySIM accepts many CAD interfaces and robot languages, but has no direct interface to any robot controllers. The support of user defined functionality is remarkable.

Finally, the ACT WELD [Aleph, 1994] the welding extension of ACT from Aleph Technologies is the only PC-based system, which specially supports welding functionalities. ACT is a CAD/CAM robot

software with extra open C interface for programming that is not usual for such systems. It is a powerful system providing not only automatic generation of the welding paths and the corresponding technology data, but also a collision free path planner for the robot's movements.

However, as none of the previous systems, neither ACT WELD uses common commercially available software tools such as the graphics modeller ACIS or the development platform AutoCAD. This is a major drawback, since the reimplementation and maintenance of basic functionalities need much more efforts (manpower and development time) as when common modules are used. Another important limit of ACT is that it supports only the low level controller (called KALI) of the developer company.

The systems differ from simple textual editors up to high performance 3D graphical environments including modules to enhance the use of the systems functionalities and the speed of program generation. But most of the functions are distributed over the different systems and none of the systems is specially designed for the use in SMEs for welding applications.

PROARC is capable of simulating [Madarasz et al, 1996], generating and reusing entire or partial welding sequences for robot program generation. The typical cell layout consists of a robot and a positioner. Torch cleaning and changing stations are also supported and the corresponding motions are automatically included in the welding sequence. The workpieces may have single or multiple seams and may be shifted individually. Technology macros have also been defined for fillet welds and V-type welds. Finally, PROARC is capable of importing different CAD formats as well as communicating with various robot controllers via an MMS (Manufacturing Message Specification) interface [MMS, 1987].

In the following the approach of the PROARC system will be discussed. First, the goals and the structure of the system are shown where the program generation and the communication modules are introduced in more detail [PROARC, 1995]. In Chapter 6 the demonstration environment and its components are briefly explained, and, finally, some conclusions are drawn.

2. PROARC Objectives

The above examination of the available programming systems allowed to draw some conclusions which were useful for the development of the PROARC system.

The target group is clearly that of SME , which surely cannot afford so high investment costs as big companies do. The costs of the required computer type, additional hardware and software had to be kept low in order to encourage companies' investment in new technologies. Personal computers and standard software are important preconditions to this goal. Furthermore, the use of well -known software tools for the realisation of a new system enables the user to work with the new system effectively after a very short time of practicing. The user will accept the system faster and the productivity will increase with the user's motivation.

On this account the AutoCAD system has been chosen as the development platform for the PROARC system. It meets the precondition of a frequently used software in SMEs, and provides the required functionalities for developing a user- friendly programming environment. E.g. the user should be freed from complicated input sequences and should be able to provide his knowledge of a welding expert for the system in a simple way. Thus, the PROARC system is able to concentrate on the application specific features. Some basic capabilities of the system are similar to those of ACT WELD. However, the major difference lies in the flexibility of the seam's programming in PROARC. No system is known, which allows a flexible restructuring of the welding sequence and the unconstrained back (modelling) and forth (simulation) stepping within a single programming session. As a conclusion of the market survey and the target SMEs the following requirements were defined for the system:

In modelling:

- support of standard CAD interfaces (IGES, DXF);
- 3D solid modelling of the entire workcell, including robot, positioner, torchstation, and cleaning device;
- support of collision checking during simulation;

In programming:

- support of Fillet and Butt welds;
- support of multiple layers;

- support of straight and circular weldlines as well as combination of these;
- automatic generation of welding paths including the appropriate welding data;
- unconstrained programming of welding sequences;
- effective access to welding entities;

In program generation and communication to the robot:

- support of a common interface between the programming and program generation part of the system to enable the use of different robot languages;
- the DNC (Direct Numerical Control) connection support of an MMS interface between the programming system and the robot to provide more control over the welding process for the user.

3. System Architecture

The system can be divided into different functional modules: Modelling, Welding Planning, Program Generation and Communication. In this division of the system the different phases of an off-line programming session are mapped. However, the underlying structure is not that separable, especially not for the first two modules.

For the development of the system's kernel, an object oriented approach has been made. Object oriented software development has a number of advances over the conventional top-down approach. Some of them are modularity, information hiding, polymorphism, which often result in a better ability for the maintenance and reusability of the software. In this project the Booch notation [Booch, 1994], has been chosen for the development. It can be used during the entire cycle of software development (analysis phase, design phase, and implementation phase). The three phases developed in an iterative way using the concurrent engineering methodology. That means, some parts of the software might be in the phase of implementation while some other parts might just have been analysed. Thus, taking another view of the system leads to Figure 1 which describes the software structure of the system.

This was the motivation for, on the one hand, the modular design of the problem domain classes and, on the other hand, the introduction

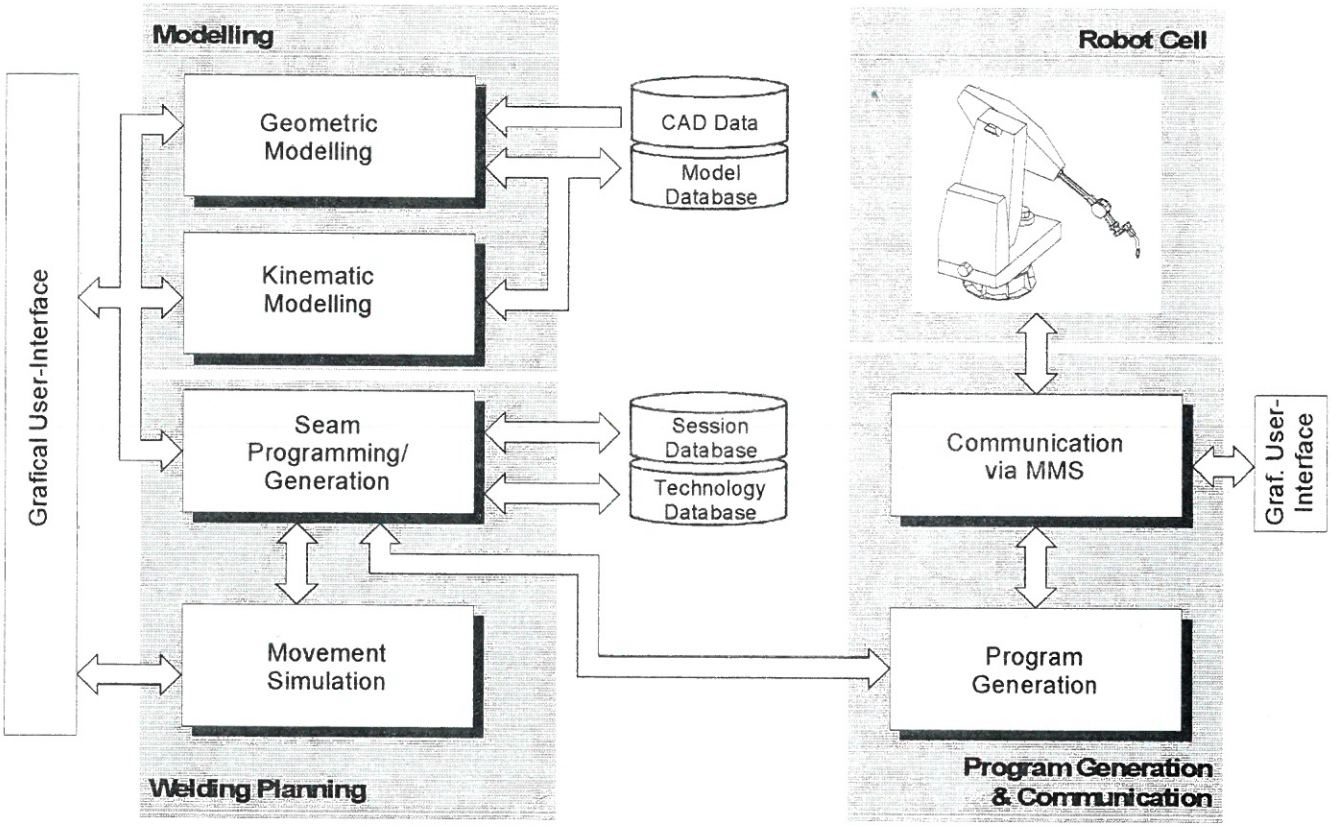

Figure 1. Principle Software Structure of the System

of a further tool to implement the user interface. This tool, the Microsoft Foundation Classes (MFC) is supplied by the Visual C++ compiler and enables the programmer to create very efficiently a Windows driven application framework. Thus, the connection to AutoCAD is reduced to the management of the graphical representations of different entities. This connection is established by a set of classes specific for the AutoCAD software. Finally, for the persistent storage of the system's runtime objects, serialisation functionalities are used, which again are supplied by the MFC.

The Modelling Module of the system provides the model of the welding environment to test for possible collisions. There are two fundamental components of the welding environment: the workpiece (with fixtures), and the entire workcell. All components of the system are modelled geometrically as solids. The workcell consists of the following items: robot, positioner, fixtures, and torches. In addition to this, the system provides models of cleaners and torchstations for auxiliary processes during a welding session (e.g. torch's cleaning/changing). Libraries of the most common types of these items will be included in the system along with the possibility to create user defined components. The model used in the PROARC project also includes some other components, the obstacles (e.g. the ceiling or the floor). The robot and the positioner are modelled as a kinematics machine with joints and links [McKerrow, 1988]. The workpiece is modelled as a whole. This means, the geometry of the workpiece is modelled in its welded state when all of the parts are joined together. In this case the designation of the seams is done by picking edges on the surface of the workpiece. At present three different torch types with three different geometries are supported in the PROARC system.

In the Welding Planning Module the user is able to program seams on the 3D model of the workpiece interactively. Usually, the layers of each seam are welded directly one after another. This sometimes results in a bad quality of the seam, due to the heat flow in the workpiece, etc. Contrary to existing off-line programming systems for welding applications, the system does not only provide the processing of single seams, but it allows the user to access each layer of a seam individually. Furthermore, the user is able to define his/her own sequencing of layers and thus the welding experience is used at this phase of the programming. Furthermore, macros have been introduced, which enable a fast and efficient posing of the welding torch and association of information technology with the seams. In all states of the program it is possible to simulate the movements of the robot and the optional positioner. Thus, the user is able to jump back and forth between the programming and the visualisation phases. This gives a maximum controllability of the welding process to the user.

The programmed welding session will finally be translated into the robot language and then downloaded to the robot. This is supported by the latter, the Program Generation/ Communication Module, which will be discussed in more detail in the following Chapters. The interface to the Program Generation Module is defined in STEP standard "EXPRESS" language. Since no predefined or standardised interfaces for the exchange of welding data exist, new STEP entities (exchange data types) were introduced. These entities (see Figure 2) apply to the demand of being so general, that they may cover the capabilities of a large number of robot languages.

The interface is a sequence list of operations where four types of operations were defined (positioning, torch specific movements, off-line search and welding):

```
ENTITY process_sequence;
        label:      OPTIONAL STRING;
        sequencelist:       LIST    [3:?]    OF
operation;
END_ENTITY;
ENTITY operation
        ABSTRACT SUPERTYPE OF
        (ONEOF(offline_search,torchspec_moves,
welding,positioning));
        operation_id:       INTEGER;
        name:       OPTIONAL STRING;
END_ENTITY;
```

At this moment two torch specific operations were introduced: torch changing and torch cleaning. All the operations are built up from lists of elementary movements. They can be simple, search and/or welding moves. These movements give the interpolation and the co-ordinates and joint values of the robot and the optional positioner as well. The most complex move is performed during a welding movement when the end of a seam is searched at the same time. All off-line searches have a predefined sequence of movements supporting the different search cases. There are welding attributes

belonging to a complete seam which is one welding operation:

END_ENTITY;

and to each of the elementary welding movements:

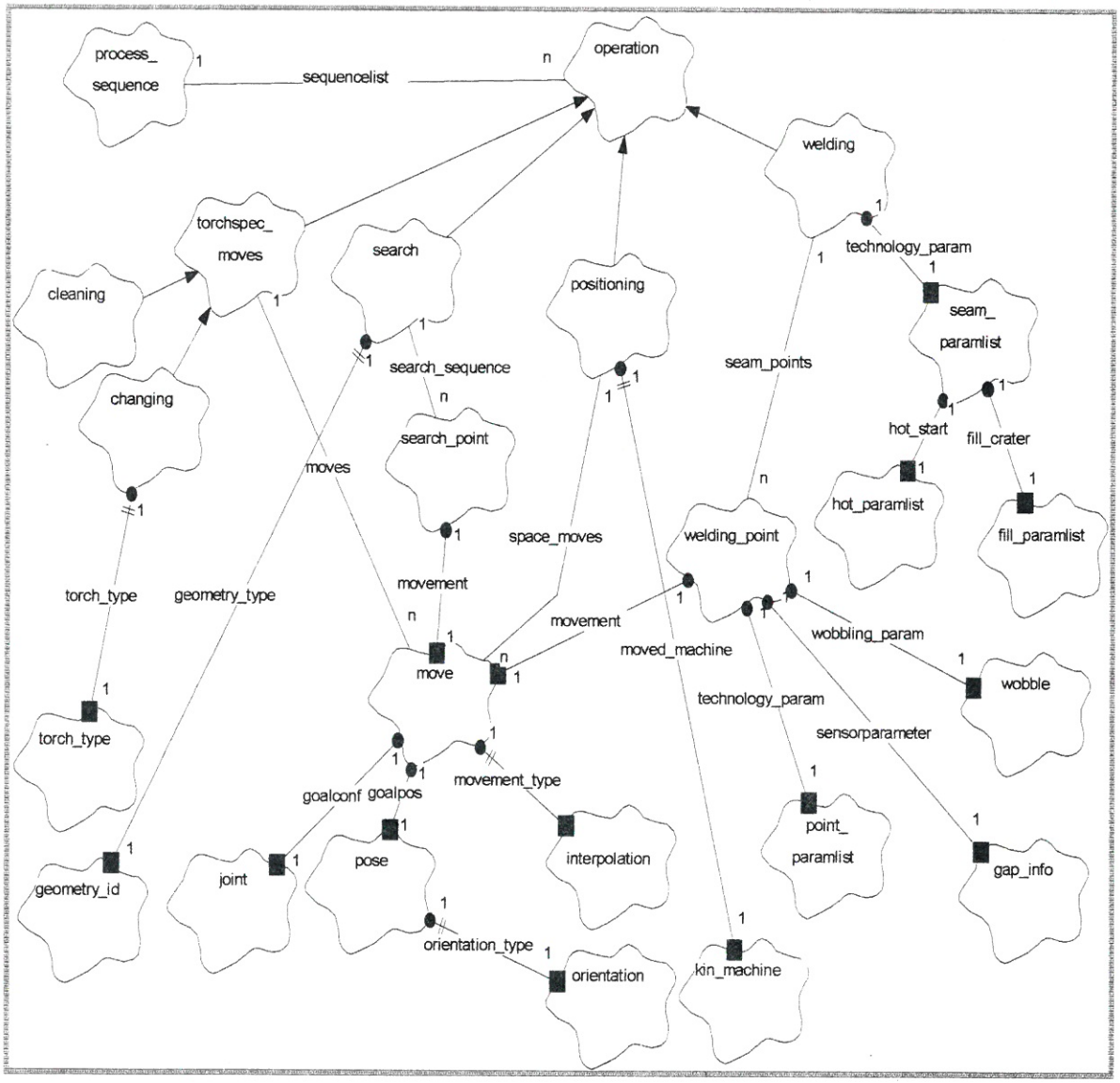

Figure 2. Booch Notation Based Class Diagram Describing the STEP Interface

ENTITY seam_paramlist;

initial_gas_time: REAL;
ending_gas_time: REAL;
hot_start_on: BOOLEAN;
hot_start: OPTIONAL hot_paramlist;
fill_crater_on: BOOLEAN;
fill_crater: OPTIONAL
fill_paramlist;
END_ENTITY;

ENTITY hot_paramlist;

crater_length: REAL;
wiremore: REAL;

ENTITY wobble;

wobblecurve_type: REAL;
wobblefrequence: REAL;
wobbleamplitude: REAL;
rest_time_right: REAL;
rest_time_left: REAL;
END_ENTITY;

ENTITY point_paramlist;

voltage: REAL;
base_current: REAL;
wirespeed: REAL;
END_ENTITY;

In order to get a maximum control of the robot while working on the off-line programming system, an MMS interface has been implemented. At a first development step mainly the basic communication functionalities are supported. However, beside this MMS connection a common DNC connection has also been established. This additionally enables basic communication to most of the robot controller types.

4. Program Generation Module

The main function of the Program Generation Module is the compilation of the STEP based robot movement description including the welding features in the actual arc welding robot language with a format, which can be directly downloaded to the robot controller. This means, the module has strong dependencies on the actual robot environment and has to be modified if applied to a different robot controller.

In most off-line robot programming systems only a limited number of robots or robot languages are supported as a final output of the system, and often no intermediate code is available to develop by the system integrator or the user has his own compiler to his own robot environment. The defined STEP description ensures the easy adaptability and reusability of this task.

However, the internal structure of the module supports the programming effort. So this module is implemented in an independent program to allow the exchange of it with a similar one if another application uses another environment.

The off-line programming session defines the entire sequence of movements for the welding of a workpiece. However, there will be deviations between the generated positions and the real positions, e.g. due to calibration errors or inaccuracies of the real workpiece. Therefore a set of search movements has been specified, which determines the real position of the workpiece by using a tactile sensor. These movements are defined for different workpiece geometries.

Figure 3 shows the 'search' points generated for the search of a vertical plate. The first point is the approach point, which the search starts on. A vertical movement and a horizontal movement determine the error in the X- and Z-direction. Finally, a movement to the front of the plate specifies the remaining error in the Y-direction. Additionally, searches are supported for horizontal plates, corners, and open geometry at the beginning as well as corners and open geometry at the ending of a seam.

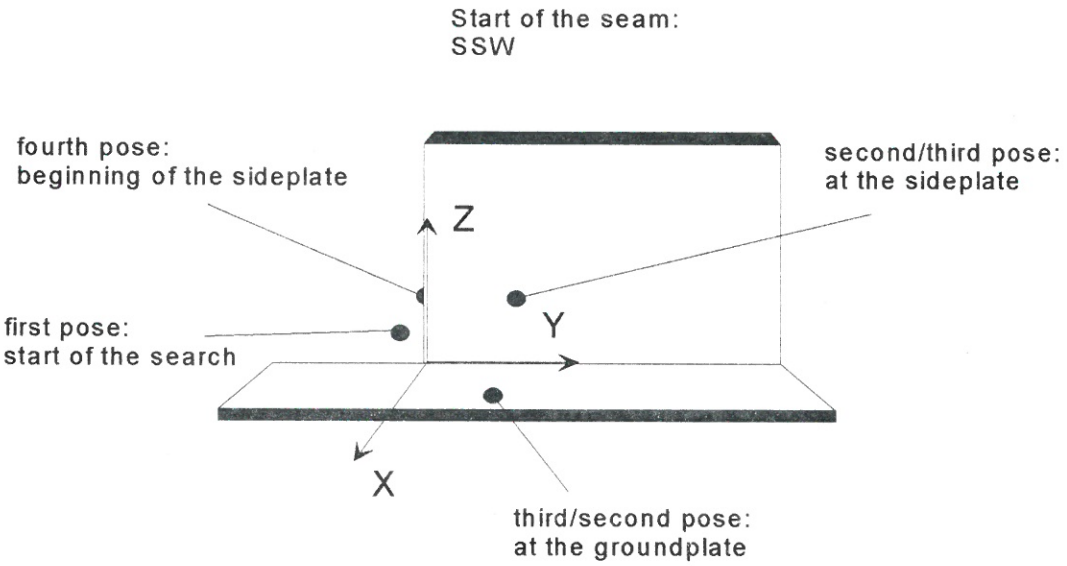

Figure 3. Search Sequence for the Search Type "Search Side Wall" (SSW)

The robot dependent parts of the program generation task can be summarised in three points. The first one is the language of the robot, the second is the format of the up- and downloading and the third one is the robot interface of the welding source, how the features of the welding can be reached from the robot.

The creation mechanism (see Figure 4) of the robot programs was developed by step- by -step

algorithms splitting up the different operations. Each operation is described by objects which consist of STEP program statements together with given robot programming language code parts. Some mappings have, of course, variable parts (e.g. co-ordinate values, delays, wirespeed). Some additional features of the program generation module were also defined to give some comfort to the user. They are general features, independent of the given environment and can be used in other versions of the program for other robots, too.

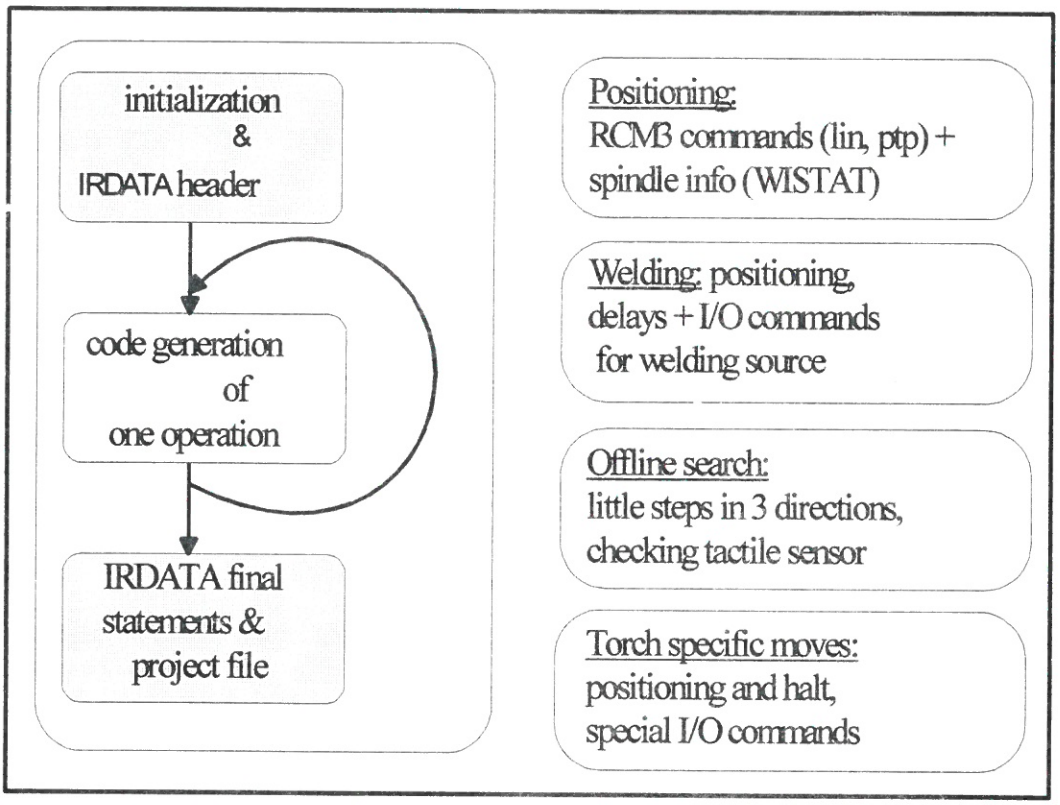

Figure 4. Creation Mechanism in the Program Generation Module

All the parameters of the program generation are stored in a so -called 'task' file. The user is able to generate it automatically and all the modifications are written directly to the file. The task file contains not only the input and the output of the program generation (STEP file and robot program), but also special switches (for example welding on, search on, etc.).

Different types of robot programs can be generated from the same source STEP file. The 'welding off' switch allows to check the exact path of the robot during the real movement without the welding itself. Three execution types of searches are defined. Programs can be generated with skipping searches ('search off'), with checking only the correct positions ('skip seam group on') and using the 'grouping' of the search and welding moves the robot may change its path according to the result of the searches ('skip seam group off'). It is possible to set the accuracy of the search.

If any errors happen during the robot program generation the user gets a note, and the generation stops. Three types of errors may occur:

- Errors in file management (e.g. unable to open the file)
- Errors during the processing of the STEP file (e.g. STEP 'entity' not found), in that case the user is informed what STEP line has problems.
- The data of the STEP file contain any erroneous information (e.g. the given speed value is out of the robot range or the given interpolation is not allowed on the actual movement).

The user can take a look at the files within the program, but the direct editing of them (either the STEP or the robot program files) is not allowed. It was decided because of safety reasons, but it can be changed easily if requested.

The modules of Program Generation shown in Figure 5 are linked together to get a single executable (EXE) file, but the modification of

them to a single dynamic link library (DLL) would be easy to do. In the following the modules' structure and their tasks will be summarized.

the PROARC STEP specification, so it could be used as a preprocessor of any STEP based files. The PROARC specific definition is given in the 'proexpr' part of the module.

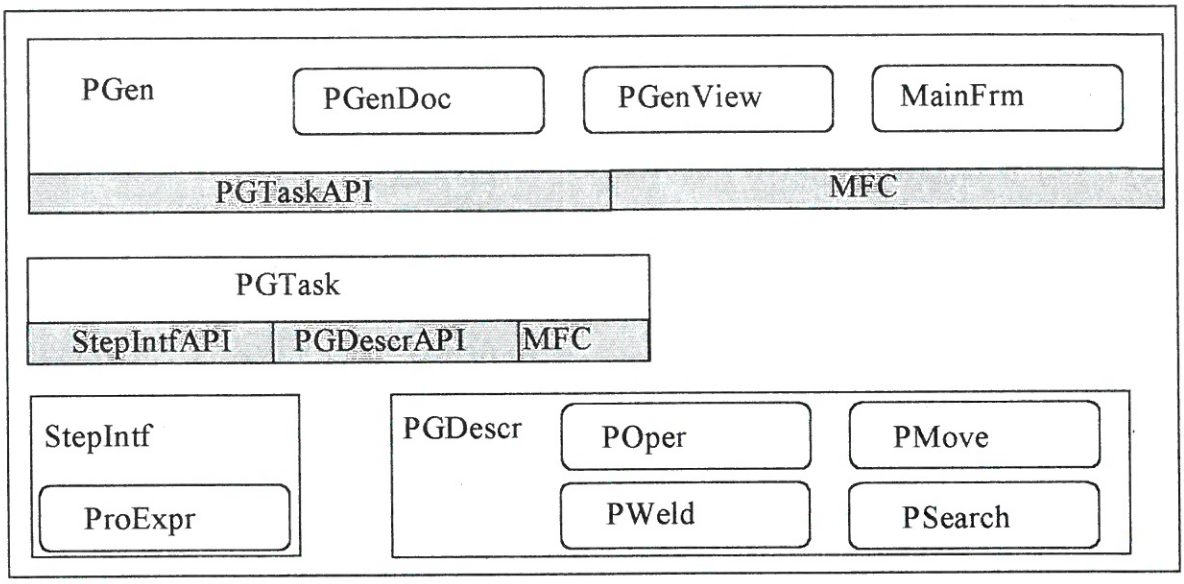

Figure 5. Software Modules of the Program Generation

PGen

This part handles the user interface of the Program Generation. The frame of this module and the three related ones were generated using the Application Studio tool of MFC. All the Dialog Boxes and one Menu were defined within the tool. The default Dialogs are used for file opening and saving and some others allow to set different values (e.g. (re)edit the task file data, or change some search options). As an extra tool with the simplest 'view' technology of MFC it is possible to look at any STEP or robot program file.

PGTask

It manages all the operations of the different files (STEP, robot program and task) in three classes. The program generation function is built up from an ordered set of these operations. This set contains general file management and special operations dedicated to a given class (e.g. load non data parts of the STEP file, update the task file with the new search options). This module contains an additional class which describes the actual robot (e.g. the existence of sensors, I/O channels belonging to different welding features).

StepIntf

This module translates the entity based STEP description into an internal representation of the program and checks the different errors. It has a special structure which is independent of

PGDescr

The operations of the classes of these modules are doing the real program generation work. The classes are used to define how the different elements of the robot program should be generated. They were built up to give a robot independent structure, where only the contents of the classes should be rewritten for another robot program. The present code pieces within the operations are developed specially for the demonstration environment. Pmove is responsible for the elementary movements, Pweld generates the welding specific parts, Psearch creates search subroutines within the robot code and Poper has different routines belonging to every operation defined in the STEP specification. There are additional parts within PGDescr generating the frame of the robot program (header and footer parts of the robot program).

5. Communication Module

Beside the programming of welding tasks, another common problem is the communication between the off-line programming system and the robot itself. The following main set of functions may be used during the remote control session in increasing complexity:

- program download/upload, delete; remote start/stop,

- determination of robot status (idle, loading, executing, etc.); read/write different variables
- create/delete variables; performing event handling
- multi-access by multiple hosts

Most of robot controllers provide only optional DNC connections via serial line and typically allow only a limited number of operations via that channel (e.g. download/upload). This serial connection has two main disadvantages:

- It obviously means a point to point connection; so it is not possible to access or monitor the robot from multiple computers.
- The speed of the communication is low.

Users want to quickly connect robots to their network in a 'client-server based' manner. They prefer low-cost and vendor independent solutions, which are consequently standard based networks instead of proprietary ones of a single vendor. These connections signify very different remote interfaces of the robots. As an alternative, the introduction of a standard based network solution such as MMS via Ethernet offers good capabilities and with the implementation of special Network Interface Units (NIU) to the robots even the multi-access and the common interface problems can be solved.

So looking for a general interface for the PROARC system towards the robot controllers, the MMS protocol and interface were chosen. It provides all the functionalities discussed above. More and more robot vendors offer such an interface, and there are proven applications where MMS interface was successfully implemented for a non MMS robot controller [Haidegger and Kuba, 1995].

The functions are used mainly by the Domain and Program Invocation (PI) Services of the MMS. The PI services allow the robot control functions: start, stop, continue of the robot program and enable the user to choose among different programs on the controller storage medium (create PI). The domain services provide the up- and download of selected programs.

This module is a Client on the MMS network exchanging messages with the robot's MMS interface. Because of the present situation, i.e. none of the industrial controller vendors supports the extensions of MMS, which is necessary for the Robot Companion Standard, its original model had to be re-designed in order to make it applicable to the real cell. The MMS Client based on a Virtual Manufacturing Device (VMD) model of the robot was specified and implemented following the robot companion standard of MMS to provide a robot independent general 'Robot VMD'. In the PROARC project additional MMS objects were introduced to get information about the status of the welding equipment.

The connection can be established with an initiation message and closed with a conclude message. Both communicating nodes, the PROARC station (Communication module) and the robot (more precisely its NIU), can skip the connection with an abort message.

Some status actions were built in to allow that the user gets information continuously from the robot. It was implemented because, from a networking point of view, the robot was not active, so all the information came from it via message poling. It is important, because without it, the Communication Module will not know if

the robot program ends, if manual torch cleaning or changing is necessary or not.

The menu list in Figure 6 shows the actions the user can activate to work with the robot. The first group of actions supports the direct remote control operations (using the MMS Program Invocation services), the second group manages the robot programs within the controller (MMS

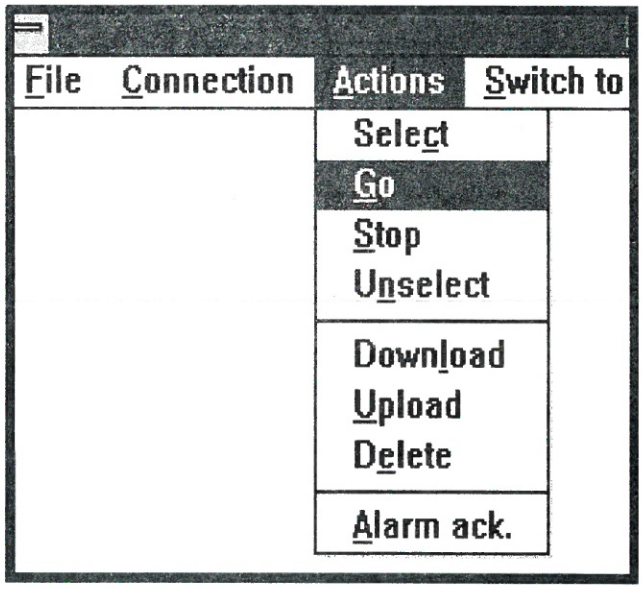

Figure 6. The Actions Menu in the Communication Module

Domain Management services). There is a further possibility (via an MMS Variable called R_ALARM) for acknowledging the alarms of the confirmations and the MapMust contains other necessary elements required by the MMS-EASE API (e.g. initialisation, error handling).

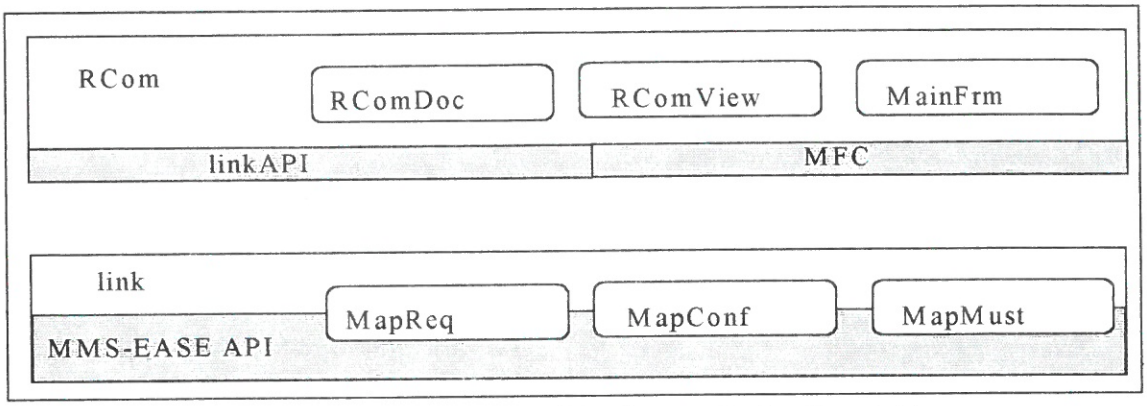

Figure 7. Software Modules of the Communication Executable

the robot controller (e.g. "Transmission Error") and the warning messages programmed in the robot code (e.g. "Change the torch" or "No workpiece found").

If the user wants to work here also with the task files created in the Program Generation module, then he/she gets the download file directly from the task file data. If an earlier uploaded version of the same robot program exists, then a warning message will force the user in choosing which one to download. Similar warning will appear if the user wants to upload the same robot program again. During one control session only one open task file is allowed.

The structure of the Communication Module (see Figure 7) shows that it was designed to be an independent program running under MS-Windows. It is also based on the MFC objects. The file management is also supported by the MFC. For the MMS communication the most commonly spread out Application Programming Interface (API), the MMS-EASE, is used [SISCO, 1994].

The modules of the Communication are also linked together to have a single executable (EXE) file. The software structure is rather simple: the user interface and file management parts (Rcom) are connected via a general API link to the MMS-EASE parts of the program. The structure of Rcom is very similar to Pgen of the Program Generation, but, of course, the functionality is different.

Link

The API link, developed in a previous project, gives a general interface for remote control of different robots. In the link part the MapReq collects the network request calls, the MapConf

6. Demonstration Environment

For the demonstration of the project's results an arc welding robot was needed to supply an MMS interface, able to communicate with the PROARC system. It was not possible to find such a robot in Hungary, so it was decided to develop a MAP interface to an arc welding robot that has a serial DNC connection. An RCM3 KUKA robot was chosen with a welding equipment from the ESS GmbH (Figure 8) in the workshop of the Automation Institute of the Kando Kalman Technical College, Budapest, Hungary.

A NIU module has been developed, which connects the RCM3 robot to the MMS network. This NIU is an MMS Server communicating with the MMS Client of the Communication Module. It sends the MMS messages to the RCM3 via its LSV2 based DNC protocol. The robot depending part, the functional mapping of the general VMD model and the given demo environment, was also developed for the NIU.

The demonstration environment of the project did not support all the features provided by PROARC. For instance, only one torch (type 0 degree) was available and some welding features were not used (e.g. arc sensors). Only a simple tactile sensor was implemented to be able to demonstrate the different search modes and sequences.

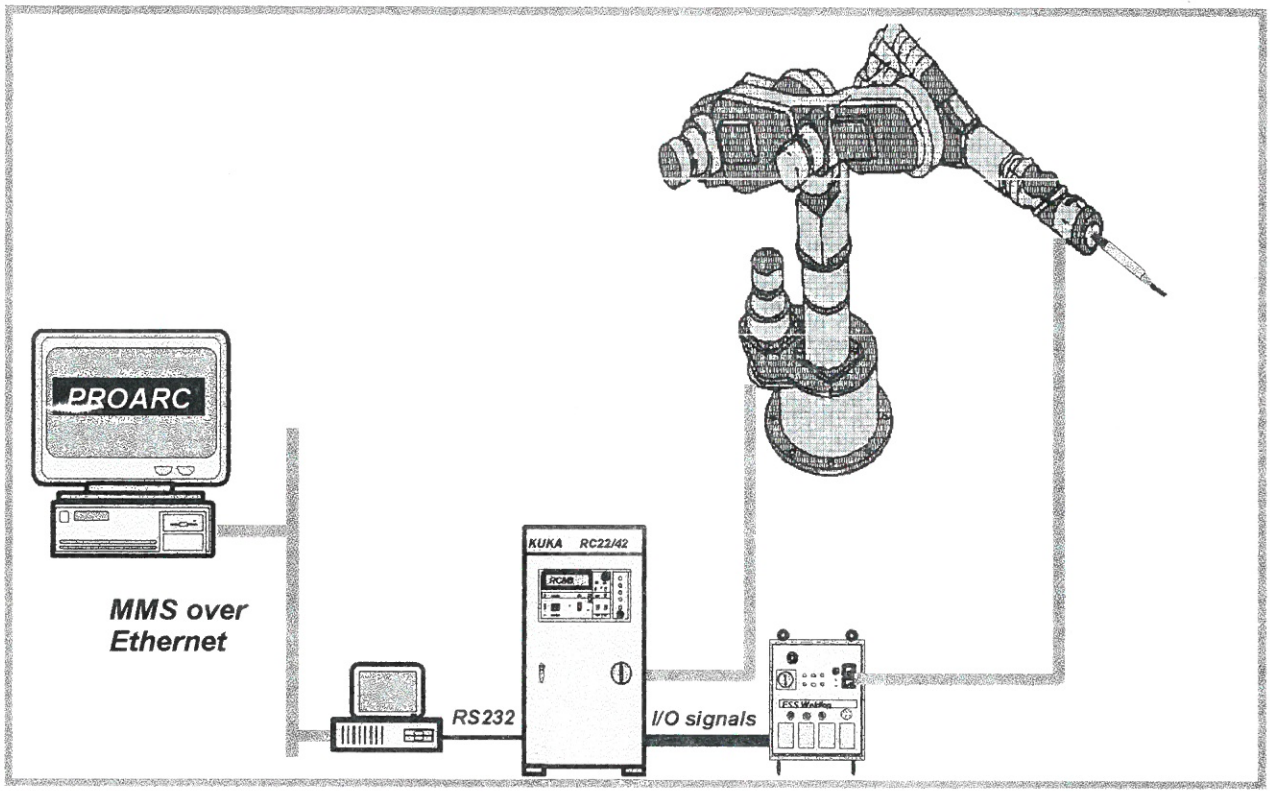

Figure 8. Final Demonstration Environment of the PROARC Project

7. Conclusions

It has been shown that the commercially available off-line programming systems for arc welding applications do not cover the needs of SMEs with small batch sizes and with one-of-a-kind production to a sufficient extent. Factors such as low costs, simple usability, and a well known software basis, on the one hand, and effective functionalities, reusability of welding programs, and an enhanced communication with the robot, on the other hand, have strongly influenced the development of the proposed PROARC system.

The joint COPERNICUS project was a success for both the Western (WZL, RWTH Aachen) and the Eastern (University of Veszprém and Computer and Automation Research Institute, Budapest) partners, as we learnt to work together effectively based on personal discussions and electronic communications, using equivalent software and hardware tools, influenced by the conclusions of the review meetings organised by the EU.

During the development we had to solve several interesting problems such as integration of different commercial and home-made software modules. Some results might be useful independently of PROARC, too. For example building of the robot inverse kinematics into AutoCAD, implementation of our general robot link interface into the communication module, etc.

As a final result of the project we reached a good prototype solution, which should and could be transformed into a useful product if some more further co-operations and financing were managed to solve open problems, as for example a more sophisticated simulation of the robot movements in AutoCAD.

The PROARC system was demonstrated in Budapest on the occasion of the final review meeting (May, 1996) where Hungarian SMEs (2-3 possible software distributors and 8-10 end users candidates) and several academic people were present. With some end users, further discussions are going on. In June 1996 the PROARC system was demonstrated in Aachen at RWTH with international participation and with a quite strong professional interest in both the software and the welding aspects.

Acknowledgments

This work is partly supported by the EC within the framework of Copernicus'93 (Reference is

PROARC No. 7831). The authors are grateful to the partners, the WZL of RWTH Aachen and University of Veszprém, for their contributions.

REFERENCES

ALEPH TECHNOLOGIES, **ACT Weld Offline Programming Software Package for Arc Welding Robots**, advertisement paper, 1994 http://www.inria.fr/RII/aleph-eng.html

BOOCH, G., **Object-Oriented Analysis and Design with Applications**, THE BENJAMIN/CUMMINGS PUBLISHING COMPANY, INC., 1994.

BYG SYSTEMS, **GRASP**, advertisement paper, 1994, http://www.bygsys.co.uk

BYG SYSTEMS, **The Use of Simulation To Design and Program Robot Welding Systems**, Internal Report, 1995.

HAIDEGGER, G. and KUBA, R., **Highly Automated Flexible Assembly System**, Proceedings of Automation '95 Conference, Budapest, Hungary, August 1995, pp. 347-353.

ifp GmbH, AnySIM-robotics, advertisement paper, 1994.

MADARASZ, L., HOLECKO, P. and RUDAS, I.J., **Robotics Systems Simulations**, Proceedings of Robotics in the Alpe-Adria-Danube Region (RAAD'96) Conference, Budapest, Hungary, June 1996, pp. 219-222.

MCKERROW, P.J., **Introduction to Robotics**, ADDISON-WESLEY PUBLISHING, INC., 1988.

MMS: **Manufacturing Message Specification**, Service Definition and Protocol, ISO 9506-1 9506-2, EIA Standard - 511, 1987.

PROARC CONSORTIUM, **Deliverable 2.1, Evaluation and Detailed Concept**, ESPRIT No. 7831, 1994.

PROARC CONSORTIUM, **Deliverable 2.4, Specification of the Program Generation and the Communication Module**, ESPRIT No. 7831, 1995.

SISCO, **MMS-EASE User Guide**, 1994, http://www.sisconet.com

Workspace, 1996, http://www.rwt.com/