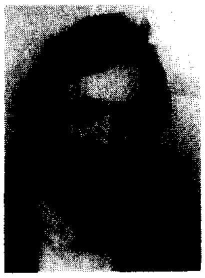
A Software System for the Simulation of Robot Based Manufacturing Processes

R. Dillmann and M. Huck
Institut für Informatik III, P.O. Box 6980, D-7500 Karlsruhe, Federal Republic of Germany

A simulation system is described which supports programming of robot based manufacturing processes. A CAD system is used for geometric modelling. The system allows modelling of different object classes like robots, end-effectors, sensors, workpieces as well as the robot's environment. Control functions on different control hierarchies in the manufacturing system can be emulated. The emulation of the virtual robots is displayed graphically. Analysis and evaluation methods are applied to detect errors and conflicts in the simulated manufacturing process with the purpose of optimizing the program.

Keywords: Software system, Simulation, Robot based manufacturing processes, Interactive programming of robots, CAD-modelling, CIM integration



Ruediger Dillmann is Research Assistant in the Computer Science Department of the University of Karlsruhe, Federal Republic of Germany, where he is head of the CAD/CAM/robotics group. Since 1978 his interests have centered on intelligent robots and graphical simulation methods. Dr. Dillmann is an internationally known expert in his field and the author of many articles and two books. He is engaged in the European ESPRIT Project and a member of CIM Europe. Dr. Dillmann received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from the University of Karlsruhe.

Martin Huck is a Research Associate in the Computer Science Department of the University of Karlsruhe. He passed his Engineer of Computer Science Examination in October '84. His research interests are: Off-line robot programming, and Integration CAD/CAM Robotics.

1. Introduction

Graphical simulation systems for the support of off-line robot programming are being increasingly devised on the basis of CAD modelling software. An interface arises here between CAD and robot programming, as provided by CAD/CAM/robotics integration. CAD modelling programs are used in order to model the geometry of robots, machine tools, conveyor systems, tools, workpieces, in brief the whole of the manufacturing installation. The kinematics, the dynamics and the important basic functions of robots, of sensors and peripherals must be emulated in the simulation system, hence the processing task and the manufacturing program can be developed as closely as possible to reality, and free from errors.

Fig. 1 shows the various ways of incorporating the simulation process in the programming of robots in manufacturing cells. The simulation of the manufacturing can be incorporated on various levels for the validation of the manufacturing program. The degree of detailing in the simulation increases from the shop floor level to the robot arm level. In the simulation at arm level elementary functions and operations in the arm control and sensor control are of interest, whereas on the cell level the interaction between machine tools, robots, peripherals and conveyor systems is investigated functionally.

The technique of emulation of control functions, i.e. the virtual imitation of control elements by means of software is a tool for the support of the design of automation systems [1]. With the computer one can for example emulate virtually the control of a manufacturing cell strictly in accordance with the prototype without having the intended hardware actually present. The user software can then be developed and tested in this emulated system. At the moment there is work in progress on graphical simulation systems as a definite part of robot programming. All the systems use CAD based geometry description systems [2–5]. Commercial CAD/robotic systems like for example McAuto Place from McDonnell-Douglas

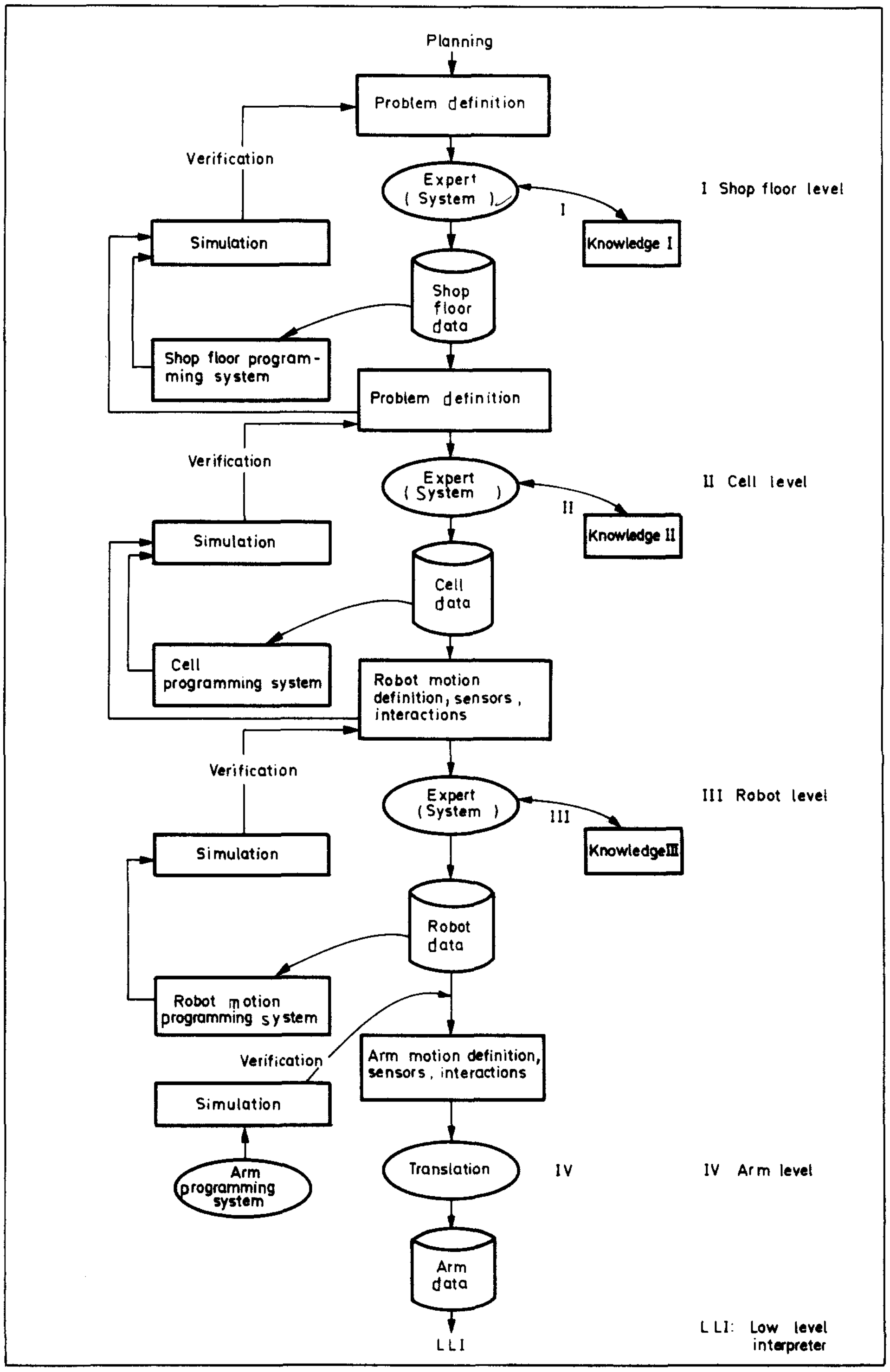


Fig. 1. Flow chart of simulation methods for the support of the programming of manufacturing tasks on various control levels.

[6] or CATIA-robotics from IBM (developer: Dassault) [7] point to a vast range of functions, like collision recognition, optimizing, dynamics, and a library of robot movements.

User-friendly and problem-oriented robot programming can be achieved by using artificial intelligence [8].

In the Department of Computer Processing Technology in the University of Karlsruhe robot programming languages have been developed for several years in combination with computer graphics [9]. ROSI 1 (Robot Simulation System) [10] is one system already in use, which supports basic operations of geometric and kinematic modelling of robots, peripherals and materials handling. The programming is done by a command language. ROSI 2 is a completely new system which has a broad spectrum of modelling software (geometry, kinematics, dynamics, sensors, object relationships), emulation software, and programming possibilities. The simulation part of the system includes analysis programs which analyse conflict situations, and alternative programs (optimizing) are offered. Furthermore there is an ability to simulate errors in order to test the reaction of adaptive sensor controlled control strategies to environmental faults or deviations. The structure and the components of ROSI 2 are described and discussed in the following sections.

2. Structure of ROSI 2

The whole system has a modular and extendable structure. Fig. 2 shows the system structure, and the individual system components chosen for function specific aspects.

The dialogue module forms a comfortable user interface via which the user can communicate with the systems by means of textual and graphic interaction techniques. Corresponding to the system components the dialogue functions are divided into function groups and define specific dialogue modes.

The modules which are accessible via the dialogue are all built according to the same plan. The core of the module concerned is a library of programs and program building blocks, which prepare the methods for the execution of the module specific tasks [11]. In addition each module has a methods monitor for the control of the execution of the commands specified by the dialogue. The methods monitor calls up individual program blocks or combines several into methods. The coupling between methods and data is effected by central data management. Before the detailed description of the individual modules in the next sections they are briefly named below.

The modelling module (M) does the physical and functional description of the objects in the environment. The CAD system Romulus is installed as a tool for the geometric modelling of the robots, tools, workpieces and sensors. The CAD data structures are extended by non-geometric data in the description of the kinematic and dynamic properties of robots and mobile conveyor devices, of the functional properties of sensors and axis control circuits, and the relationship between objects in their environment.

The emulation module (E) comprises the methods which are necessary for the planning and emulation of the robot's behaviour for simulation purposes. On-line planning, the function of the different sensors and the control and decision functions are emulated. New methods for planning and programming systems are to be continuously incorporated into this module for implicit robot programming.

The programming module (P) offers the programmer various methods of program development. The textual and graphical specification is supported by program statements. An experimental mode permits the experimental and development testing of individual line sequences and elementary operations, also their inclusion in robot programs. The purpose of this module is the preparation of a simple useful and problem oriented programming interface.

The simulation module (S) is used for the validation and verification of the programs generated. The robot program is presented in the modelled environment on the graphics display unit. Several analysis methods allow the programmer to analyse and evaluate the program sequence. Very complex processes can be investigated by means of expert functions. Errors, sources of error, or critical states identified by the simulation methods are presented on the GDU or printed out alphanumerically.

The graphics module (G) facilitates the evaluation of the simulated sequences. It offers methods with the help of which the user can increase the

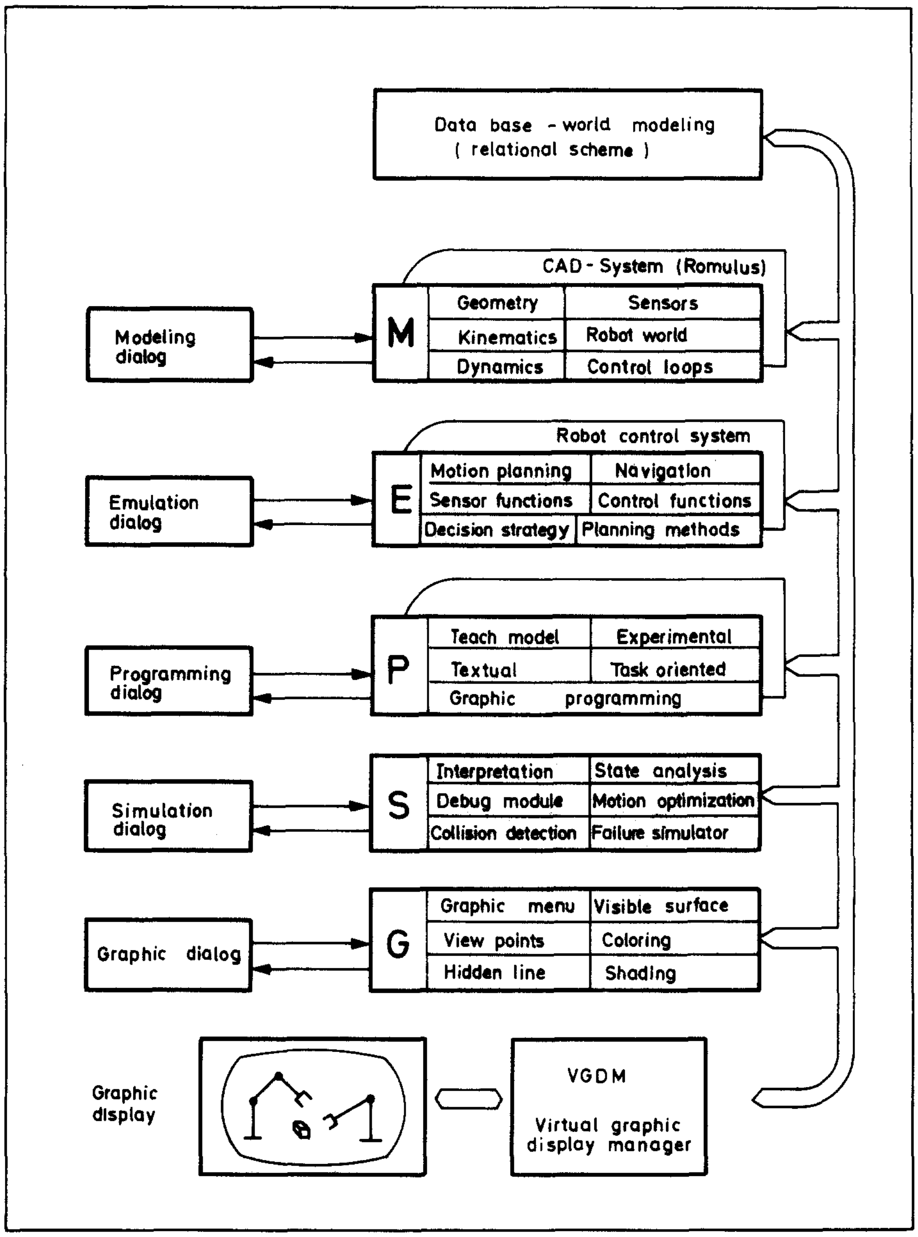


Fig. 2. System structure and components of ROSI 2.

information content in the graphics presentation. For example hidden line procedures improve the visual presentation of the third dimension, or several viewpoints of a scene permit a more exact analysis.

The interface between the simulation and the graphic systems is realized via a virtual-graphic display manager (VGDM). The VGDM makes the system independent of equipment. It offers for simulation purposes pseudo-graphic commands for the construction and manipulation of the image structure. Each module can effect changes in the graphic presentation via these commands. Specific equipment dependent drivers present the pseudo-display file stored in the VGDM on the equipment specific function structure.

As shown in Fig. 2 each module works with the central data management. In addition the methods and data are managed separately, thus ensuring consistency in the data to be processed. The data management uses a relationship data model based on RODABAS [12]. Objects of the same type are assembled into object classes. Each object class posseses a structure described by attributes. Relationships between objects are described by attributes whose values represent object names. Some examples are given in the following sections.

3. The Dialogue Frame of the System

The dialogue frame performs essentially two tasks as a communication link between system and user.
(a) It gives the user textual and graphic communication via a dialogue terminal and the incorporated graphics system.
(b) It analyses the input and transfers the parameters necessary for the execution of the function to the corresponding methods monitor.

The dialogue has a function independent and table controlled command decoder for the textual command input [13]. The command syntax and parameter sequence is described by table entries. The tables are read in the initialisation phase. Hence change and extension of the command languages is easily effected.

A number of input techniques (menu, window technique) and various graphics input equipment (dial, mouse, lightpen, pendant) are offered for the graphics command and parameter specification. By using the graphics input line points can be specified (with the lightpen), joint movements (dials) inputted, or the objects to be grasped (lightpen) are chosen. The specification of motion types for the movement, the choice of elementary robot actions and, with corresponding intelligence of the emulation, task oriented actions are possible via the choice of menu or by means of function keys.

A formula dialogue is offered to the user as a further means of communication, which creates object class specific formulas for the definition and manipulation of objects in the robot's environment. These methods offer to the user a comfortable and user oriented communication interface. The requirements of variously qualified users are also considered by the use of automatic system information which can be called out [14].

Table 1 gives the module specific dialogue modes and some typical interaction modes and the relevant selectable methods.

Table 1
Dialogue modes, interactions and methods choice.

Dialogue mode	Methods	Interactions
Modelling dialogue	CAD modelling sensors, environment relationships, functional and technological object data	Command language, menu Formulas
	Dynamics	
	Kinematics	Formulas, graphics (dials)
Emulation dialogue	Emulation of functions	Command language
	Chaining of functions and definition of the data flow	graphically controlled dialogue in the function netting
Programming dialogue	Textual programming	Command language (textual)
	Teach model	Function keyboard, dials
	Experimental programming	graphics (dials, lightpen) menu, function keyboard
	Goal oriented programming (in workpiece coordinated)	Menu, graphics (dials. lightpen)
Simulation dialogue	Choice analysis and evaluation functions	Command language, menu
	Debug	
	Error simulator	Command language
Graphics dialogue	Methods choice	Command language, menu

4. The Modelling System

The modelling system fulfils two overall requirements:
(a) Generation of the layout for robot configurations, robot manufacturing cells, or whole assembly lines.
(b) The input, description and automatic processing of all data of the physical and functional objects and the structured storage of this data in the data management. The totality of this data creates the environment model as a central core on which all other system modules are based.

Physical objects in the environment include

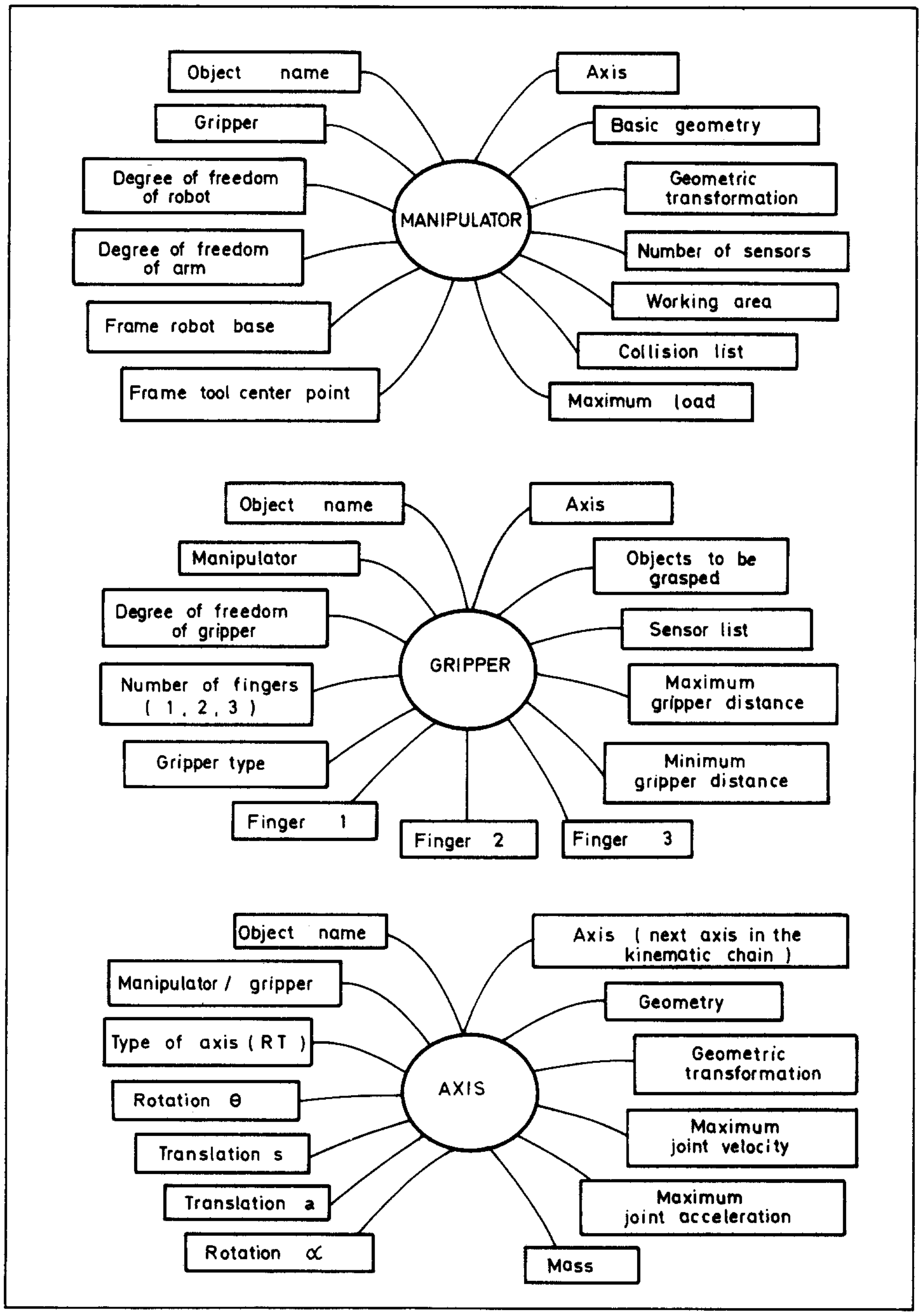


Fig. 3. Object class structure of manipulator, gripper and axis.

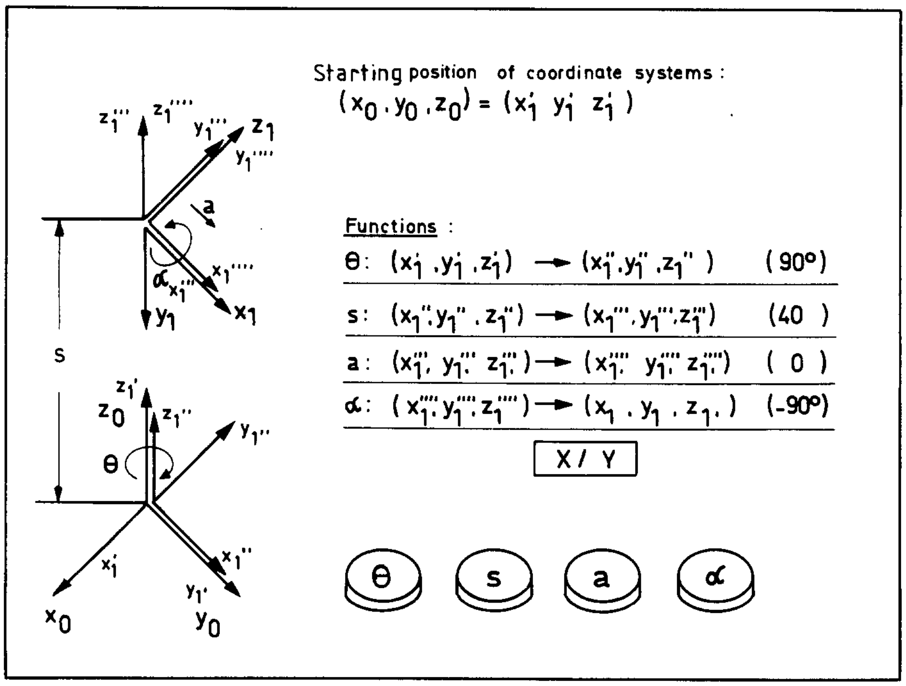
robots, conveyor devices like belts or vehicles, workpieces to be processed and assembled, and machine tools. The geometry has to be described, and for moving objects the kinematics and dynamics, and the technical features of the workpieces. By the term functional objects is understood sensors, control circuits, and relationships between objects and their environment, in which sensors are characterised by their geometry and principal dimensions. The environment model results from the objects stored in the data base, whose structures are given by the object classes concerned.

The structure of the object classes [manipulator], [gripper] and [axis] from which robots are constructed, is shown in Fig. 3 [10]. Each object class is characterised by the number and the types of the attributes to be described. Object classes for workpieces and conveyor systems are implemented in accordance with the same model. The description of functional objects is based on the object classes [sensor], [servo-loop] and [connections] (relationship to the environment). The object class concept is open and can be optionally extended by other object classes.

The CAD package Romulus is used for the geometric modelling. Romulus permits the modelling of exact geometries and creates volume models internally. Romulus also computes geometry dependent data like volume and centre of gravity, which are important for the dynamics as components of the internal geometrical representation. Each object class in the description of physical objects refers via the attribute-geometry to the internal geometric representation, which is likewise stored in the data management.

The kinematics of the robots are described by the parameters of the coordinate transformations of robot coordinates into the environment coordinates (Cartesian coordinates), and reversely. The Denavit-Hartenberg matrix method [15] is used for the transformation of robot coordinates into environment coordinates. This method is universal and can be uniformly used for any robot configuration.

Fig. 4 shows the method by which the programmer can specify the four parameters by graphic input using dials, which give the relationship between the movement coordinate systems of two adjacent arm elements [10].



Starting position of coordinate systems :
$(x_0, y_0, z_0) = (x_1' \ y_1' \ z_1')$

Functions :

θ:	$(x_1', y_1', z_1') \rightarrow (x_1'', y_1'', z_1'')$	(90°)
s:	$(x_1'', y_1'', z_1'') \rightarrow (x_1''', y_1''', z_1''')$	(40)
a:	$(x_1''', y_1''', z_1''') \rightarrow (x_1'''', y_1'''', z_1'''')$	(0)
α:	$(x_1'''', y_1'''', z_1'''') \rightarrow (x_1, y_1, z_1)$	(-90°)

Fig. 4. Graphical input of the transformation parameters.

The transformation from environment to robot coordinates cannot be described by a homogeneous method and must be computed robot specifically. This coordinate transformation is stored together with the robot description in a robot library.

As necessary parameters in the integration of the dynamics in the emulation besides the quantities calculable from the geometry, there are to be added to the environment model inertia tensors, friction parameters and the kinematics of the centre of gravity of the masses concerned.

Fig. 5 shows the internal model of a modelled PUMA 600 robot. Generated objects form the basis of the model, which are linked via the kinematics description to the typical sequence of movements.

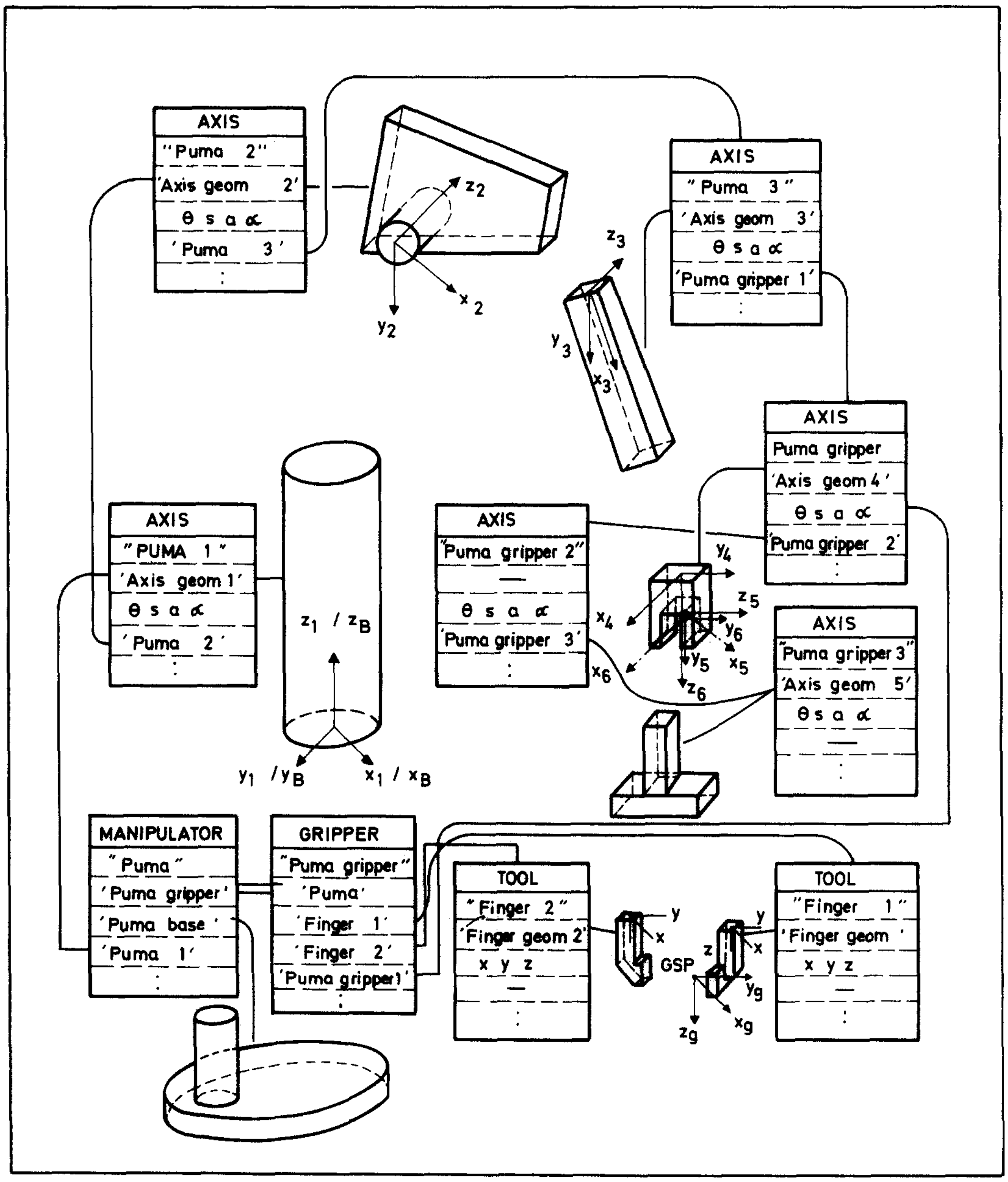


Fig. 5. Internal model of a PUMA 600 Robot.

Each object refers to its geometry, which is represented visually.

The objects defined as functional are described in the modelling by their typical parameters. Descriptions of the type of servo, the servo-parameters and the servo-loop control structure are necessary for the servo-loop simulation. These quantities can be specified by previously defined formulas.

Relationships between objects in the environment can be described by objects in the object class combination. An object combination is defined by a distinct name, which includes the names of the objects present in the relationship and which describes the relationship itself. Relative to two objects a relationship can describe a temporary combination by the simple placing of one after another, or a rigid combination in the case of the screwing together of the two objects. Likewise included in the environment modelling is the positioning of objects in the environment (manufacturing cells) relative to an environment coordinate system.

The integration of sensor signals in the emulation of robot programs requires the model specific description of the sensors. A possible structure of the sensor object class is shown in Fig. 6. This structure is suitable for the class of sensors which for reasons of their operating principle make possible the evaluation of the sensor signal from the geometric conditions in the environment. Representatives of this class include proximity and tactile sensors. The attributes of the class sensor refer to the mounting of the sensors in the robot, the type of sensor, the signal form and any signal threshold of the sensor itself. Further attributes describe the principle of operation for the evaluation, e.g. geometric relationship between straight line (sensor signal) and surface (workpiece surface) and the operator (emulated signal), e.g. the distance between attachment point to the intersection between straight line and plane. A handling sequence can describe the further actions dependent on the sensor signal, a sensor macro is used for the emulation in the choice of relevant methods, and for the graphic simulation in the visualisation of the sensor signal.

Other methods must be developed for sensors that cannot be emulated by geometric relationships, e.g. the production of tables of sensor signal values, whereby from the evaluation the constraints included in the table are to be evaluated, too. Fig. 7 shows the concept of sensor emulation in a modelled environment [16]. On one side are all environment objects in their internal object class specific description without the robot. The

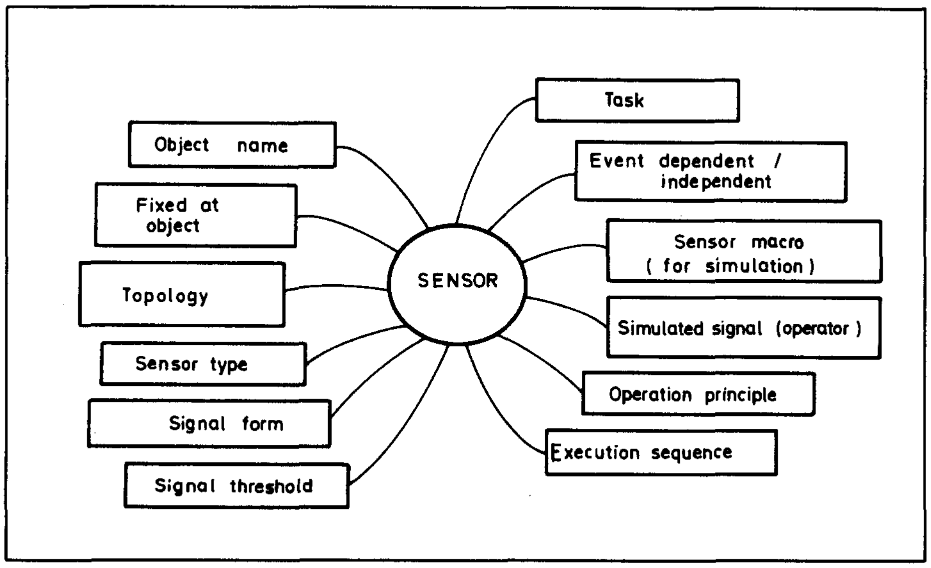


Fig. 6. Structure of the object class sensor.

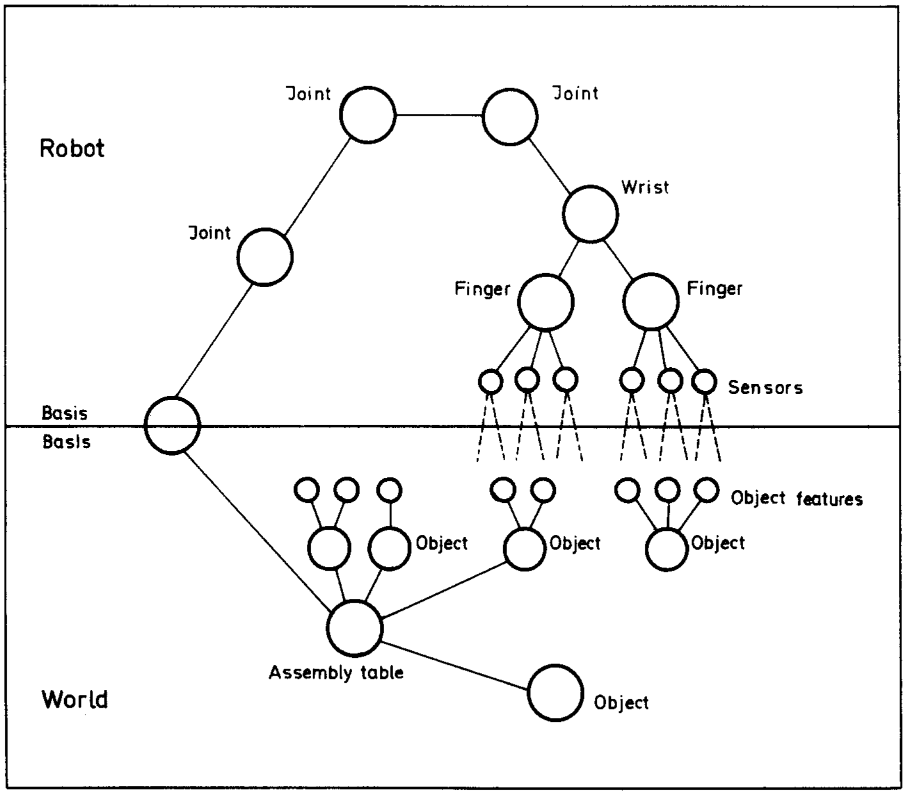


Fig. 7. Principle of sensor emulation for the environment model.

robot is modelled in the same way and is equipped with sensors. The sensors are described according to the method shown in Fig. 6. These data forming the environment model make up in their totality the basis of sensor emulation, in which the methods for the evaluation of the corresponding geometric relationships or the tables are presented as algorithms.

5. Emulation of Functions

The control and the behaviour of a robot are characterised by very many interconnected functions. Fig. 8 shows a hierarchised control architecture [17] which includes elementary basic functions which interact both horizontally and vertically. The behaviour of the control can be simulated by software reproduction (emulation) of these functions. Components in robot control, like for example:

- line planning
- collision avoidance
- offset switching
- coordinate transformation
- adaptive sensor controlled line planning
- servomechanisms for the drive axes
- gripper control
- sensor functions
- sensor processing
- logic control functions
- manipulation of internal data
- navigation functions etc.

can be emulated. In their entirety they virtually represent a robot. Each emulated function is formally characterised by its input and output data, its internal data, and its behaviour. An interpolator has for example the parameters of the desired track as input data (polynomial type, initial, final and ancillary conditions), and the interpolation frequency. The joint angle statements of the interpolated track values and error indications are output data.

Emulated basic functions can be joined in sequence, i.e. the output variables of a function are the input variables of the next function. The re-

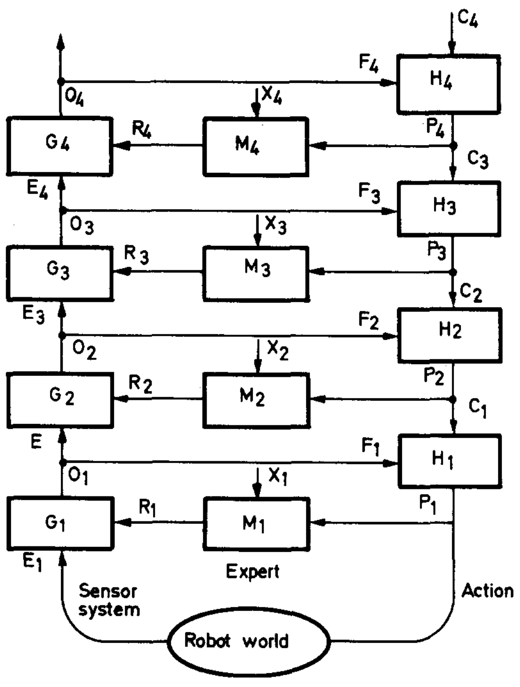


Fig. 8. Emulated control functions and their interconnection. H = control operators; M = actualisation and management operators for the environment model; G = sensor operators.

sulting data flow represents the control of the robot, or parts of it. The emulation can therefore be regarded as a prototype for the actual system. The design and the development of robot controls or cell controls [17] can in this way be developed before the actual hardware is available.

The solution of the equation of motion must be emulated for the investigations into the dynamics of kinematic chains. Wittenburg and Wolz [18] have developed the MESA VERDE program for this. In the control of the equation of motion the servomechanism laws have to be emulated and combined with the equation of motion. Emulated sensor functions permit the reproduction of sensor supported adaptive servo-loops or complex multisensor structures. Logic functions, decision strategies and learning and optimizing algorithms can likewise be emulated.

Fig. 9 shows the schematic structure of a control level within a hierarchical control system. All the blocks and their linkage and the input/output interfaces can be emulated.

A graphically executed dialogue supports the method design in the emulator. Further the coordination of the modelled object data to the designed methods must be monitored. A method monitor is superimposed on the dialogue which looks after this task. When the emulation methods have been linked with the appropriate data objects, the programming can be applied to the emulated system.

6. Programming

The user can choose his desired program mode via the program dialogue. The following program modes can be chosen:
- textual programming
- graphical programming (teach-in)
- experimental programming
- programming in the debug mode.

The programming is divided into separate levels corresponding to its complexity. There are specific programming levels oriented to the hierarchical composition of the robot control (also in the manufacturing cells). Thus the following levels are distinguished:
- complex task command (implicit statements)
- complex function command (explicit statements)
- movement command
- motion planning parameter
- trajectory command.

The corresponding emulated robot functions can be programmed and implemented via these program levels. Movement instructions activate for example the line planning methods (line parameter formulation, interpolation, and coordinate transformation). Grip commands (general effector commands) operate on the emulated gripping functions. Tracking commands (relative movements) refer to the emulated kinematic properties of conveyor belts, rotating tables, or moved objects. Sensor commands and corresponding branching commands make the programming of adaptive control strategies possible.

Another command group supports the alphanumeric specification of frames and reference points.

Textual, graphical and experimental programming can be basically executed on all levels. SRL [19] and IRDATA [20] in ROSI 2 are used as programming languages for the simulation of robot applications. An extended command language is

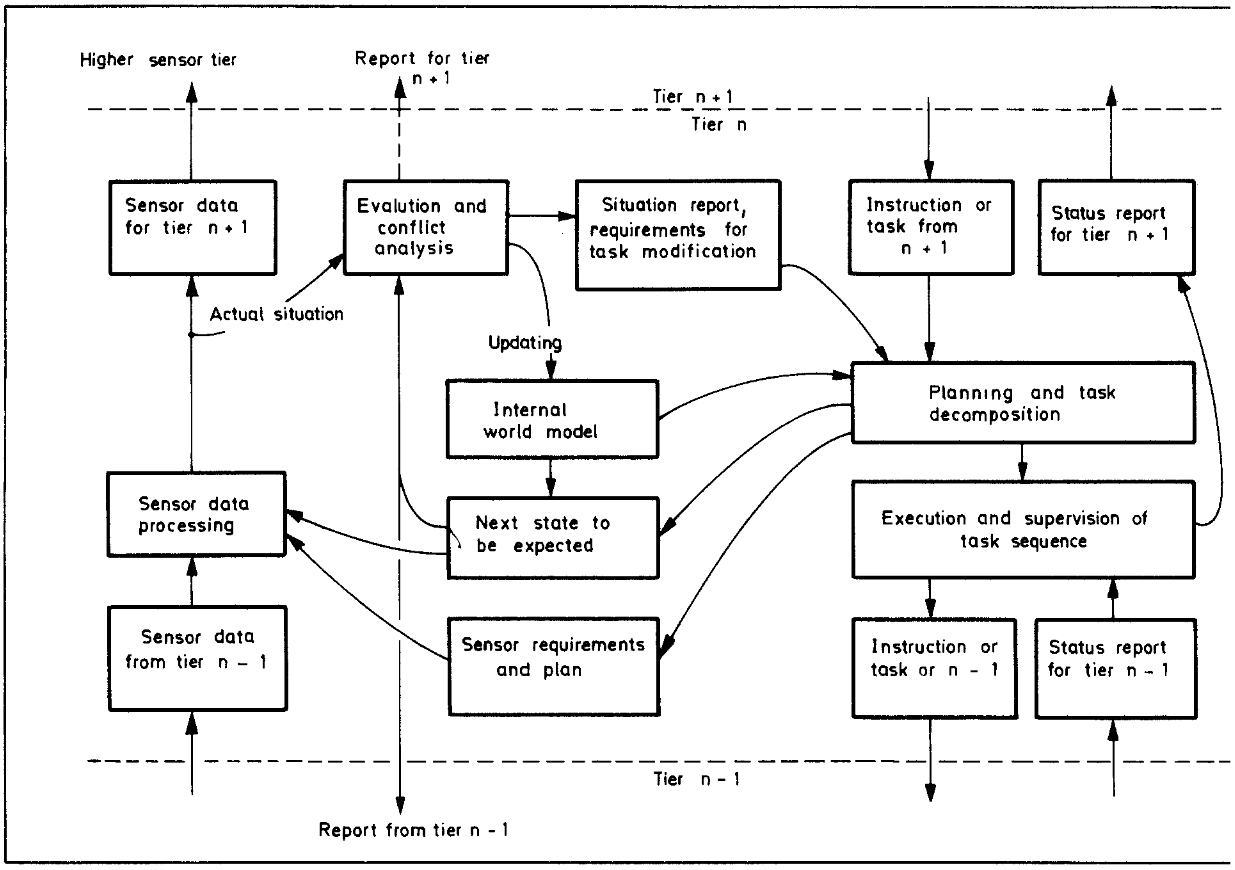

Higher sensor tier
Report for tier n + 1
Tier n + 1
Tier n
Sensor data for tier n + 1
Evalution and conflict analysis
Situation report, requirements for task modification
Instruction or task from n + 1
Status report for tier n + 1
Actual situation
Updating
Internal world model
Planning and task decomposition
Sensor data processing
Next state to be expected
Execution and supervision of task sequence
Sensor data from tier n − 1
Sensor requirements and plan
Instruction or task or n − 1
Status report for tier n − 1
Tier n - 1
Report from tier n - 1

Fig. 9. Detailing of the functions within a control level.

implemented in the programming and simulation of elements and functions. The commands called up are individual functions or sequences of functions. Tables 2 and 3 show some commands in the command language. Graphical programming is done in the dialogue via the visual display unit interface. Movement tracks, actions, changes of status in frame and reference points are defined with mouse, pendant, or lightpen. The experimental programming is done by using the statistically visualised environment of the robot (or the manufacturing cell) interactively with the immediately simulated execution of the command on the GDU. Alphanumeric data of the trajectory, internal states, collisions, etc. are presented to the programmer. The programmer can program at will in Cartesian, in robot, in tool, or in relative coordinates. Also the higher functions like "GRASP", "INSERT" or "MOVE AROUND OBSTACLE" can be used in this mode.

The debugger is an important programming mode for the support of the program verification. The already described SRL or IRDATA programs can be modified as desired in the debug mode. The parameter list is either modified in the command concerned, or the whole command is modified, erased, or a new command is issued.

7. Simulation as a Tool to Support the Formulation of the Program

The object of the simulation by using the methods of the simulation module can be characterised by three general statements.

(a) The graphical presentation together with the analysis and debug functions of the simulation module is intended to help the programmer to test and validate his program intelligently.

(b) Interpretation procedures and evaluation func-

tions permit the programmer to optimize syntactically and semantically correct programs as regards efficiency.
(c) Newly developed planning procedures for implicit planning, e.g. planning of fine movements in the assembly or collision avoidance algorithms can be integrated into the emulation module. The evaluation in the simulation helps to reveal errors of weaknesses in new methods.

A series of functions implements fulfilment of the specified goals, from which the programmer can build up the interpretation methods required. The interpretation permits in parallel with the emulation the evaluation and registering of the chosen actions, e.g. the registering of the joint actuators, the gripper position and orientation, or the monitoring of collision spaces. If the emulation of the sensor cannot be represented graphically, the programmer can issue by means of alphanumeric output information about the sensor, time of the measurement and the sensor signal value. The emulation can proceed in real time, or with a time delay. The debug mode permits step-by-step program emulation, makes possible the incorporation of breakpoints and trace functions. The quantities which are to be registered in the trace are specified by the programmer. Breakpoints can be incorporated in order to check the states at the break by evaluation methods for conflict or possible optimizations. The conflict analysis has to provide methods for the evaluation of different processes, e.g. gripping and assembly processes, collaboration between several arms or optimum approaches to objects.

Intelligent procedures are to be implemented for the realisation of these simulation methods which apply to the work envelope model and a knowledge base. The knowledge base must pro-

Table 2
Simple commands in the programming of robot movements.

Command	Meaning	Parameter
Movement to frame		
MFR	in robot coordinates	frame BTYP, time
MFW	in world coordinates	frame BTYP, time
MFE	in effector coordinates	frame BTYP, time
MFP	in piecepart coordinates	frame, BTYP, time + name (piecepart)
MFC	on circle	frame, ZWPKT, time
Movement to position		
MPR	robot coordinates	position, BTYP, time
MPW	world coordinates	position, BTYP, time
MPE	effector coordinates	position BTYP, time
MPP	piecepart coordinates	position, BTYP, time + name (piecepart)
MMO	Move according moving object	name (object to be moved)
MMJ	Individual joint movement	joint number, value, time
Movement		
MAF	of an object to approach frame	name (object), time
MSL	to object, straight line	name (object), time
MAO	via approach point to object	name (object), time, $\|\|V_{AP}\|\|$
MDF	via departure frame away from object	name (object), time
OGR	Open gripper	value, time
OGM	Open in the next movement	value
CGR	Close gripper	value, time
CGM	Close in next movement	value
SON	Switch on effector	
SOF	Switch off effector	
AZS	Acquire zero setting	time
STP	Stop	time

Table 3
Command groups for specification and generation of robot program data.

Command	Meaning	Parameter
PER	Protocolling effector frame in robot coordinates	
PJA	Protocolling joint angle	
POF	Protocolling object frame	name (object)
PMV	Protocolling of the maximum value	
DFV	Define frame variable	name (frame variable), frame
DPV	Define position variable	name (position variable), position
DOM	Define object movement	moving parameter
DCM	Define conveyor system movement	name (conveyor system), v
AOC	Attach object to conveyor system	name (O), name (C), time, frame
MSC	Move robot in synchronism with conveyor system	frame, time, name (*Conveyor*)
ROS	Robot out of synchronism	frame, time
WAI	Wait	time
DHS	Define handling sequence	name (*H.S.*), sensor, name (*Pgr*)
EHS	Execute handling sequence	name (*H.S.*)
GEN	Generate robot/conveyor program	name (program), name (manip.)
END	End program	
RUN	Run robot program	name (program)
SYN	Synchronisation of several programs	
RPR	Robot program	name (program)
CPR	Conveyor program	name (program)
ESY	End synchronisation	

duce rules in order to recognize and diagnose errors. A seam forming process may be taken as an example. The conditions for the seam forming process are stored in the knowledge base. The task of the conflict analysis function is to check the data of the world model for compliance with the conditions given in the knowledge base. If inconsistencies occur in the check the errors must be diagnosed. To this end further knowledge is evaluated about the seam forming process and the registering of the last program steps. The analysis function isolates the error and helps the programmer to correct the error. The error diagnosis allows step-by-step improvement by addition of further controls.

The simulation module acts as an error simulator for the testing of algorithms of adaptive planning methods, which facilitates the deliberate introduction of errors by the programmer. Thus newly developed and implemented processes can be tested by the simulation too.

It would be best to have an automatic error diagnosis and error correction by using the methods of simulation. If this goal is reached the knowhow is ready for the generation in the planning module of the implicit programming, optimally programmed and free from errors.

The design and implementation therefore also serves the development of intelligent planning systems (expert systems) for the deployment of robots.

8. Graphical Visualisation

The following graphics system visualises the modelled environment and the actions which a robot program describes. In the analysis of critical situations and states there must be created a detailed graphical three dimensional representation which agrees with the environment used. The graphics module offers the programmer functions which are intended to improve the quality of certain graphical presentations. Stationary scenes in the cell can be analysed by means of the hidden line method. Hidden line for dynamic picture sequences and shading would support the visualisation of collisions and the presentation of the third dimension. Variable viewpoint definitions and zoom functions can be incorporated in order to highlight detailed assembly processes in the analysis.

The virtual graphic display manager (VGDM) shown in Fig. 2 forms the interface between the components of the simulation system and the graphics system used. The VGDM offers the interface in the simulation system a quantity of pseudo-graphic commands in order to build up the image structure and to execute dynamic manipulations. By the term dynamic manipulations one understands changes in the image structure which result in the emulation and the simulation. Examples of these include:

(a) displays of joint movements (operators are calculated in the movement emulation),
(b) gating in and gating out of symbols and alphanumeric information for the outputting of the results of the simulation,
(c) display of emulated sensor functions,
(d) display of the errors recognised in the simulation, e.g. blinking or colouring of colliding objects.

Further general graphic functions which the VGDM must include are for example:

(a) perspective views,
(b) zoom function,
(c) changes in viewpoint,
(d) processing of inputs via the graphics peripherals (e.g. dials, picking with lightpen),
(e) reading of the graphic data structures from transformations controlled by the graphic inputs.

The simulation system creates a pseudo-display file by using the pseudo-graphic orders. The pseudo-display file is processed by a graphics system-dependent driver program, which generates the display file for the graphics system concerned. Thus the driver program recognises any hardware implemented graphic transformations in the graphics system (e.g. perspectives), and relieves the VGDM from having to calculate transformations of this type. In order to join another graphics system onto the simulation system, only the specific driver has therefore to be implemented. Changes in the simulation software are not necessary.

9. State of the Implementation

The development work for the simulation system was carried out on a VAX 750 under VMS in the Pascal implementation language. The PS 300 made by the Evans and Sutherland Company was used for the graphics system. The graphics peripherals consist of an input unit with eight dials and a pendant in the graphics input via an indicator pen. Romulus is used for the geometric modelling which stores the objects generated in the Femgen files [21]. Version 1 of ROSI has already been put into use. It comprises dialogue functions, modelling functions, the motion planning and graphics display. Fig. 10 shows a mobile robot equipped with two arms which is located in the library and which has been given over to the PS 300.

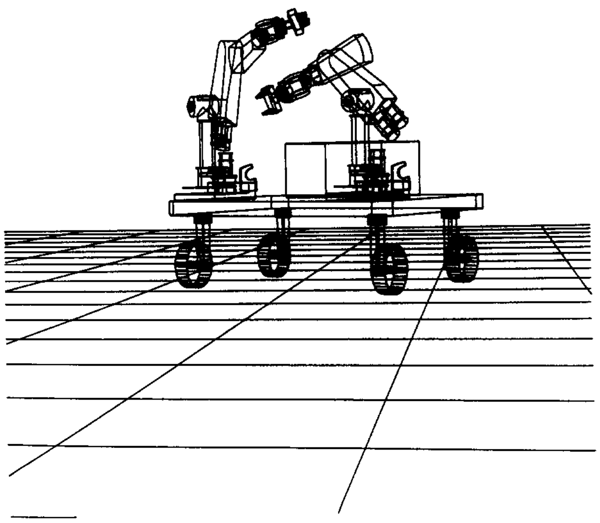


Fig. 10. 3D graphics of a mobile two-arm robot (made by G. Werling and W. Scherr).

Version 2 is extended by additional emulation functions, the simulation module and the programming interface. The installation of the VGDM is also envisaged.

After the installation of Version 2 coupling up with an actual robot will follow, in order to perform accuracy analysis and to test the simulation system under working conditions.

References

[1] Bloom, H.M., Furlani, C.M., McLean, C.R. (1984) *Emulation as a Tool in the Design of Factory Automation Systems.* Internal Rep. Factory Automation System Division, Center of Manufacturing Engineering, National Bureau of Standards, Gaithersburg MD 20899.

[2] Soroka, B.F. (1984) *Debugging Robot Programs with a Simulator.* Proc. CADCAM 8 Conf., Anaheim, California.

[3] Wesley, M.A. et al. (1980) *A Geometric Modelling System*

for Automated Mechanical Assembly. IBM J. Res. Development, 24, pp. 1–12.

[4] Sata, T., Kimura, F. et al. (1981) *Robot Simulation System as Task Programming Tool*. Proc. 11th. ISIR, Tokyo.

[5] Computer graphics for robot off-line programming 1981: Product description of McDonnell-Douglas Automation Company.

[6] PLACE: Software product description manual. McDonnell-Douglas Company, 1983.

[7] CATIA Robotic. User manual. Dassault Systems.

[8] Schmidt-Streier, U., Altenheim, A. (1984) *Geometric Data Acquisition with Computer Graphics and Sensors in the Programming of Industrial Robots*, pp. 155–159. Conference report CAMP '84, Berlin (in German).

[9] Dillmann, R. (1983) *A Graphical Emulation System for Robot Design and Program Testing*. Proc. 13th. ISIR/ROBOTS 7, pp. 7.1–7.15. Chicago, Illinois.

[10] Hornung, B., Huck, M. (1984) *Design and Implementation of an Interactive 3-D Real Time Robot Emulation System*. Diploma Dissertation, Univers. Karlsruhe (in German).

[11] Maier, H. (1984) *Management of Methods and their Coupling to a Workpiece Model in an Integrated CAD System*. VDI Specialist Report, Series 10, No. 13. Düsseldorf (in German).

[12] Müller, E., Pods, R. (1982) *Design of a Robot Data Base*. Diploma Dissertation, Univers. Karlsruhe (in German).

[13] Rupp, M. (1975) *Table-Controlled Decoding of Command Languages with Low Store Expense*. Elektronische Rechenanlagen 17, pp. 271–276 (in German).

[14] Schmitt, A. (1983) *Dialogue System*. B-1-Wissenschaftsverlag (in German).

[15] Denavit, J., Hartenberg, R.S. (1955) *A Kinematic Notation for Lower Pair Mechanisms Based on Matrices*. J. Appl. Mech. 77, pp. 215–221.

[16] Andre, G., Boulic, R. (1984) *Graphics System and Proximity Sensors for Robot Programming*. Laboratoire d'Automatique Campus de Beaulieu (in French).

[17] Albus, J.S. et al. (1981) *Theory and Practice of Hierarchical Control*. 23 IEEE, Sept. 13–17, Washington.

[18] Wittenburg, J., Wolz, U. (1985) *MESA VERDE – A Computer Program for the Simulation of Non-Linear Dynamics of Multi-Body Systems*. Robotersysteme I, pp. 7–18 (in German).

[19] Blume, C., Jakob, W. (1983) *Design of the Structured Robot Language (SRL)*. Proc. Adv. Software for Robotics, Conference, Lüttich.

[20] Blume, C., Frommherz, B. (1984) *IRDATA – An introduction*. Elektronik 25, pp. 60–66 (in German).

[21] Argyle, E.R. (1984) *Romulus to Femgen Interface*. Shape Data.