FULLY AUTONOMOUS WEB BASED VIRTUAL ROBOT PROTOTYPING AND MANUFACTURING

TAREK SOBH, RAUL MIHALI
School of Engineering and Design, University of Bridgeport, USA

ANATOLI SACHENKO
Ternopil Academy of National Economy, Ukraine

ABSTRACT

Developing an environment that enables optimal and flexible design of robot manipulators using reconfigurable links, joints, actuators, and sensors is an essential concept towards efficient robot design and prototyping. Such an environment should have a complex set of software and hardware subsystems for designing the physical parts and the controllers, and for the algorithmic control of the robot modules (kinematics, inverse kinematics, dynamics, trajectory planning, analog control and digital computer control). Specifying object-based communications and catalog mechanisms between the software modules, controllers, physical parts, CAD designs, and actuator and sensor components is a necessary step in the prototyping activities.
In this project, we propose a web interface based prototyping environment for robot manipulators with the required subsystems and interfaces between the different components of this environment. The goal is to build a system of components that allows potential customers (located anywhere geographically) to input through a web interface a set of request / design parameters and specifications (such as torque, dexterity, repeatability, velocity etc), that could be analyzed, simulated and converted to specific manufacturing information that can be ultimately used by an automated manufacturing plant. The plant would be able to build consumer robots tailored to specific requirements and deliverable to customers flexibly.

KEYWORDS: autonomous prototyping, manufacturing, web control

I. INTRODUCTION

Designing and building an electro-mechanical system such as a robot manipulator, requires diverse series of tasks, starting with specifying the tasks and performance requirements, determining the robot configuration and parameters that are most suitable for the required tasks, ordering/manufacturing the parts and assembling the robot, developing the necessary software and hardware components (controller, simulator, monitor), and finally, testing the robot and measuring its performance.
Our goal is to build a framework for the optimal and flexible design of robot manipulators with the required software and hardware systems and modules that are independent of the design parameters, so that it can be used for different configurations and varying parameters.

The task will imply various types of integrations, ranging from the development of a web interface, the design and consideration of software simulators and controllers to design of hardware controllers, parts and automated (robotized) manufacturing facilities. To make the proposed model truly accessible and marketable, the set of requirement tasks, torques, dexterity, repetability etc will be integrated into a web based portal, hence easily access through a URL. Figure 1 shows a schematic view of the prototyping environment with its sub-systems and the interface.

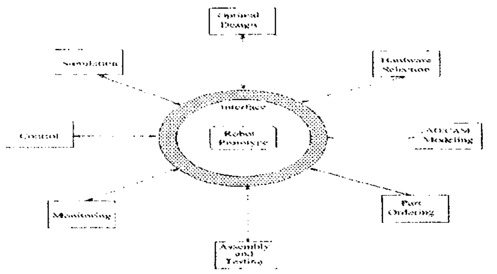

Figure 1. The Prototype Environment

II. BACKGROUND / EXISTING WORK

To integrate the work among different teams and sites working in such a large project, there must be some kind of synchronization to facilitate the communication and cooperation in between. A concurrent engineering infrastructure that encompasses multiple sites and subsystems, called Palo Alto Collaborative Testbed (PACT), was proposed in [2]. The issues discussed in that work were: cooperative development of interfaces, protocols, and architecture, sharing of knowledge among heterogeneous systems, and computer-aided support for negotiation and decision-making.

An execution environment for heterogeneous systems called ``InterBase" was proposed in [1]. It integrates preexisting systems over a distributed, autonomous, and heterogeneous environment via a tool-based interface. In this environment each system is associated with a Remote System Interface (RSI) that enables the transition from the local heterogeneity of each system to a uniform system-level interface.

Object orientation and its applications to integrate heterogeneous, autonomous, and distributed systems are discussed in [5]. The argument in this work is that object-oriented distributed computing is a natural step forward from the client-server systems of today. Automated, flexible and intelligent manufacturing based on object-oriented design and analysis techniques is discussed in [4], and a system for design, process planning and inspection is presented. A management system for the generation and control of documentation flow throughout a whole manufacturing process is presented in [3]. The method of quality assurance is used to develop this system that covers cooperative work between different departments for documentation manipulation. A computer-based architecture program called the Distributed and Integrated Environment for Computer-Aided Engineering (DICE), which addresses the coordination and communication problems in engineering, was developed at the MIT Intelligent Engineering Systems Laboratory [6]. A prototyping envirnoment for robot manipulators and a 3-link manipulator protopype have been developed atUtah (Utah Prototying Environment [UPE] and Utah Robot Kit [URK]) [7, 8]. Various controlling, simulation and optimization methods have been succesfully presented and exercised on custom manipulators such as a tire changing manipulator, a generic 6 DOF manipulator, etc [9,10,11,12,13]. The Ternopil Academy of National Economy (TANE) team has proven extensive work on controller development including development of controller structures, controller hardware and software, reconfigurable SHP and IOS [14,15,16,17,18].

III. IMPLEMENTATION CONSIDERATIONS

Overall Design

The Prototyping and Manufacturing Environment (PME) consists of a central interface (CI) and subsystem interfaces (SSI). The tasks of the central interface are to:

- Maintain a global database of all the information needed for the design process.
- Communicate with the subsystems to update any changes in the system. This requires the central interface to know which subsystems need to know these changes and send messages to these subsystems informing them of the required changes.
- Receive messages and reports from the subsystems when any changes are required, or when any action has been taken (e.g., update complete).
- Transfer data between the subsystems upon request.
- Check constraints and apply some of the update rules.
- Maintain a design history containing the changes and actions that have been taken during each design process with date and time stamps.
- Deliver reports to the customer with the current status and any changes in the system.

Consumer Preset Constraints, Task Specifications and Requirements

A database for the system components and the design parameters is necessary to enable the CI to check the constraints, to apply the update rules, to identify the subsystems that should be informed when any change happens in the system, and to maintain a design history and supply the required reports.

This database contains the following: Robot configuration, Design parameters, Available platforms, Design constraints, Subsystems information, Update rules, General information about the system.

Now the problem is to maintain this database. One solution is to use a database management system (DBMS) and integrate it in the prototyping environment. This requires writing an interface to transform the data from and to this DBMS, and this interface might be quite complicated. The other solution is to write our own DBMS. This sounds difficult, but we can make it very simple since the amount of data we have is limited and does not need sophisticated mechanisms to handle it. A relational database model is used in our design, and a user interface has been implemented to maintain this database. For the current design, by making a one-to-one correspondence between the classes and the files, reading and writing a file can be accomplished by adding member functions to each class. In this case no need for a special DBMS and all operations can be performed by simple functions.

This makes the data entry scalable. A customer can add, update, and delete any constraint or update rule through a web portal. Complicated data structures are not required for evaluation.

Design Parameters

The main purpose of this system is to keep track of these parameters and notify the subsystems of any changes that occur to any of these parameters. The design parameters are the most important data items in this environment. For the system to perform this task, it needs to know the following data: a complete list of the design parameters, and which subsystems should be notified if a certain parameter is being changed.

TABLE I. Subsystem Notification Table According to Parameter Changes

Design Parameter	CI	Design	Control	Simulation	Monitor	[illegible]	[illegible]	Ordering	Assembly
robot model	○	●	○	○	○		○		○
link length	○	●	○	○	○		○		○
link mass	●		○	○			○		○
link density	○	●					○		○
link cross area	○	●					○		○
joint location	○	●	○	○			○		○
[illegible]	●						○		○
update rate	○	●	○	○	○	○			
comm. rate	○	○	○	○		●			
motor spin	○							●	○
motor range	○	●	○	○	○			○	○
sensor range	○	●	○	○	○	○		○	○
PID parameters	○	●	○	○					
display rate	○				●				
platform	○				○	●			○

○ To be notified	● Make change

Table I shows a list of most certain design parameters along with the subsystem that can change them and the subsystems that should be notified by a change in any of these parameters. Notice that some of these parameters are changed by the CI, this change is accomplished using the update rules. In this figure note that one of the design parameters can be removed from this table, which is ``display rate." The removal of this parameter is valid because only one subsystem needs to know about this parameter and it is the same subsystem that can change it. However, we will keep it for possible future extensions or additions of other subsystems that might be interested in this parameter.

Database Design Considerations

A simple architecture for the database design is to make a one to one correspondence between classes and files, i.e., each file represents a class in the object analysis. For example, the robot file represents the robot class and each of the robot subclasses has a corresponding file. This design facilitates data transfer between the files and the system (the memory). On the other hand, this strong coupling between the database design and the system classes violates the database design rule of trying to make the design independent of the application; however, if the object analysis is done independently of the application intended, then this coupling is not a problem.

Now, we need to determine the format to be used to represent the database contents and the relations between the files in this database. Figure 2 shows the suggested data files that constitute the database for the system, and the data items in each file. The figure also shows the relations between the files. The single arrow arcs represent a one-to-one relation, and the double arrow arcs represent a one-to-many relation.

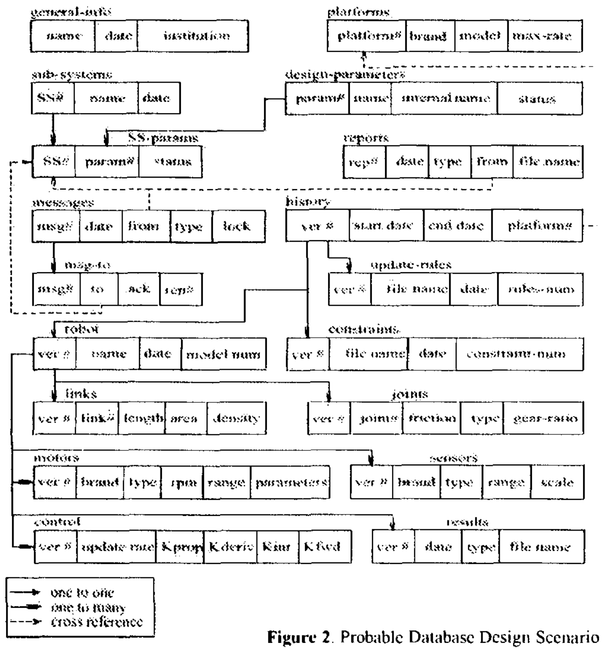

Figure 2. Probable Database Design Scenario

Controller Controller Simulator Block Design

The first step in the design of the controller for a manipulator is to solve for its kinematics, inverse kinematics, dynamics, and the feedback control equation that will be used; the type of input and the user interface should be determined at this stage too.

For trajectory generation, cubic polynomials method will be considered, which generates a cubic function that describes the motion from a starting point to a goal point in a certain time. The error in position and velocity is calculated using the readings from the sensors. The control module simulates a PD controller to minimize that error. The error depends on several factors such as: frequency of update, frequency of sensors reading, and the desired trajectory (for example, if we want to move with a big angle in a very small time interval, the error will be large).

The CSB will function in direct feedback through the CI with the customer web portal, such as to allow changes to the constraint sets until they qualify as valid for the manufacturing possibilities.

Once the adequate dataset has been analyzed and solved through the CSB, the CI will pass the relevant information to the robotized manufacturing facility (RMF).

The Robotized Manufacturing Facility

In order to completely isolate the robot manufacturing from comprehensive human operator interventions, a robotized manufacturing facility / plant is considered. The plant will be composed of robot manipulators itself, and will be responsible of assembling the parts based on the provided information from a successful CSB execution. An extensive collection of "universal" hardware parts will be available for this purpose, such as satisfactory varieties of robots can be constructed with same parts.

IV. DESIGN APPROACH

To assure consistent viability, in the early stages of the proposed project, the focus will fall on fairly simplified sets of input parameters and "output" manipulators. Specifically the designable robots will resume to limited torque, mobility and visual interface requirements, then hopefully reaching to the levels of automation / performance desired on industrial scale manipulators, based on the financial and intelectual resources and interests available .

REFERENCES

1. BUKHRES, O. A., CHEN, J., DU, W., AND ELMAGARMID, A. K. "Interbase: An execution environment for heterogeneous software systems", IEEE Computer Magazine (Aug. 1993), 57-69.
2. CUTKOSKY, M. R., ENGELMORE, R. S., FIKES, R. E., GENESERETH, M. R., GRUBER, T. R., MARK, W. S., TENENBAUM, J. M., AND WEBER, J. C. "PACT: An experiment in integrating concurrent engineering systems", IEEE Computer Magazine (Jan. 1993), 28-37.
3. DUHOVNIK, J., TAVCAR, J., AND KOPOREC, J. "Project manager with quality assurance, Computer-Aided Design" (May 1993), 311-319.
4. MAREFAT, M., MALHORTA, S., AND KASHYAP, R. L. "Object-oriented intelligent computer-integrated design, process planning, and inspection", IEEE Computer Magazine (Mar. 1993), 54-65.
5. NICOL, J. R., WILKES, C. T., AND MANOLA, F. A. "Object orientation in heterogeneous distributed computing systems", IEEE Computer Magazine (June 1993), 57-67.
6. SRIRAM, D., AND LOGCHER, R. "The MIT dice project", IEEE Computer Magazine (Jan. 1993), 64-71.
7. M. DEKHIL, T. M. SOBH, T. C. HENDERSON, AND R. MECKLENBURG, "UPE: Utah Prototyping Environment for Robot Manipulators", In proceedings of the IEEE ICRA, Nagoya, Japan, May 1995.
8. M. DEKHIL, T. M. SOBH, AND T. HENDERSON, "URK: Utah Robot Kit – A 3-link Robot Manipulator Prototype", In proceedings of the IEEE ICRA, San Diego, May 1994.
9. R. MIHALI M. GRIGORIAN AND T. M. SOBH, "An Application of Robotic Optimization: Design for a Tire Changing Robot", In TSI Press Series on Robotic and Manufacturing Systems: Recent Results in Research, Development and Applications, Volume 10, pp. 421-428, 2000.
10. TAREK M. SOBH AND ABDELSHAKOUR A. ABUZNEID, "A Generic Simulator/Controller for Robot Manipulators", in Recent Advances in Mechatronics, Springer Verlag, pp. 575-589, 1999.
11. R. MIHALI AND T. M. SOBH, "The Formula One Tire Changing Robot (F1-T.C.R.)", in proceedings of the IEEE International Conference on Robotics and Automation (IEEE ICRA 99), May 1999, Detroit, Michigan.
12. TAREK M. SOBH AND ABDELSHAKOUR A. ABUZNEID, "A Generic Simulator/Controller for Robot Manipulators", In proceedings of the IEEE 2nd International Conference on Recent Advances in Mechatronics, Istanbul, Turkey, May 1999.
13. M. GRIGORIAN AND T. M. SOBH, "Design-Simulation-Optimization Package for a Generic 6-DOF Manipulator with a Spherical Wrist", In TSI Press Series on Robotic and Manufacturing Systems: Recent Results in Research, Development and Applications, Volume 10, pp. 413-420, 2000.
14. MELNYK A., EICHELE H., POCHAEVETS A., YATSYMIRSKYJ M. Fast Orthogonal Transforms Core Architecture. Proceedings of the 43rd International Scientific Colloquium, Vol 1, Germany, 1998, p.509-514.
15. MELNYK A. DSP System Based on Programmable Processor with Scalable Parametrizable Fast Orthogonal Transforms Hardware Core // Proceedings of the XI Conference "Application of Microprocessors in Automatic Control and Measurement", Volume 1, Warsaw, Poland, 1998, P.87-98
16. MELNYK A., ERMETOV Y. Facilities of Specialized VLSI Designing from Algorithmical Task Description to Register-transfer Level. "Modern Problems of Telecommunication, Computer Engineering and Specialists Training", International Scientific-Technical Conference, Lviv, 1998, p. 108.
17. MELNYK A., EICHELE H., POCHAEVETS A., YATSYMIRSKYJ M. A Parametrizable Scalable ASIC Core for Fast Orthogonal Transforms. George Simon Ohm Fachhonchshule, Nuremberg, Schriftenreihe, 1998, p.13-22.
18. A.SACHENKO, V.KOCHAN, V.TURCHENKO, "Intelligent Distributed Sensor Network", Proc. of IMTC/98, St.Paul, USA, vol.1, 1998, pp.60-66.