Concurrent Engineering and Robot Prototyping

Mohamed Dekhil, Tarek M. Sobh, Thomas C. Henderson, and Robert Mecklenburg [1]

UUSC-93-023

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

October 1, 1993

Abstract

This report addresses the theoretical basis for building a prototyping environment for electro-mechanical systems using concurrent engineering approach. In Designing a robot manipulator, as an example of electro-mechanical systems, the interaction between several modules (S/W, VLSI, CAD, CAM, Robotics, and Control) illustrates an interdisciplinary prototyping environment that includes different types of information that are radically different but combined in a coordinated way. We propose an interface layer that facilitates the communication between the different systems involved in the design and manufacturing process, and set the protocols that enable the interaction between these heterogeneous systems to take place.

[1] This work was supported in part by DARPA grant N00014-91-J-4123, NSF grant CDA 9024721, and a University of Utah Research Committee grant. All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

Contents

List of Figures

List of Tables

1 Introduction

In designing and building an electro-mechanical system, such as robot manipulators, a lot of tasks are required, starting with specifying the tasks and performance requirements, determining the robot configuration and parameters that are most suitable for the required tasks, ordering 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 optimal and flexible design of robot manipulators with the required software and hardware systems and modules which are independent of the design parameters, so that it can be used for different configurations and varying parameters. This environment will be composed of several sub-systems. Some of these sub-systems are:

- Design.
- Simulation.
- Control.
- Monitoring.
- Hardware selection.
- CAD/CAM modeling.
- Part Ordering.
- Physical assembly and testing.

Each sub-system has its own structure, data representation, and reasoning methodology. On the other hand, there is a lot of shared information among these sub-systems. To maintain the consistency of the whole system, an interface layer is proposed to facilitate the communication between these sub-systems, and set the protocols that enable the interaction between these heterogeneous sub-systems to take place.

Figure 1 shows the interaction between some of those sub-systems. The optimal design system affects the control and the simulation systems. The monitor sub-system

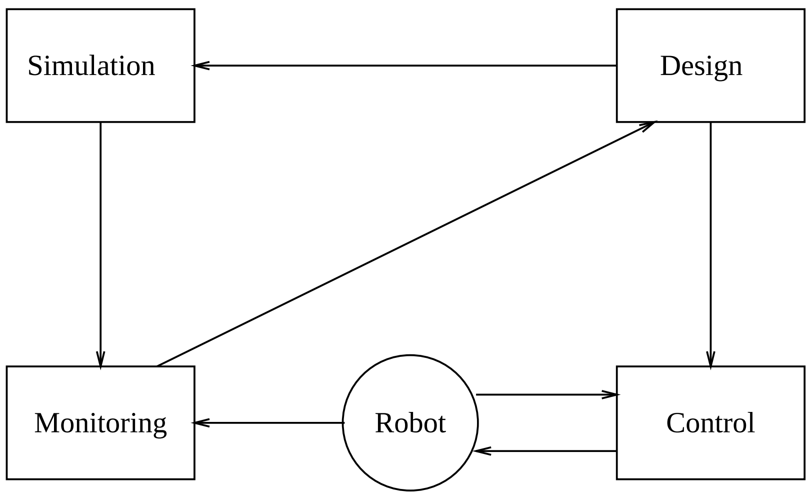

Figure 1: Interaction Between Sub-systems in the Prototyping Environment.

takes its data from the simulator and from the robot. There is also feedback information from the monitor to the optimal design system to refine the design according to the performance measurements for each design. The robot is derived by the control system, and feedback information goes from the robot sensors to the control system.

A prototype 3-link robot manipulator was built to help determine the required subsystems and interfaces to build the prototyping environment, and to provide hands-on experience for the real problems and difficulties that we would like to address and solve using this environment. More details about this can be found in [10].

2 Objectives

The objective of this research project is to explore the basis for a consistent software and hardware environment, and a flexible framework that enables easy and fast modifications, and optimal design of robot manipulator parameters, with online control, monitoring, and simulation for the chosen manipulator parameters. This environment should provide a mechanism to define design objects which describe aspects of design, and the relations between those objects. The importance of this project arises from several points:

- This framework will facilitate and speed the design process of robots.

- This project will facilitate the cooperation of several groups in our Computer Science department (VLSI group, Robotics group), and the cooperation of the department with other departments (Mechanical and Electrical Engineering).
- This project will establish a basis and framework for design automation of robot manipulators.
- The interdisciplinary nature of the proposed research will provide an exceptional educational environment for those involved in the work.

This report is divided into three parts: first, a brief background for concurrent engineering and heterogeneous systems is presented with the related work in this area. Second, the proposed interface layer between the systems is described. Finally, the required representations (knowledge base, object oriented scheme, rule-based reasoning, etc.), are discussed.

3 Background and Review

There are several definition for the term *Concurrent Engineering* (CE). One definition proposed by Cleetus [2] is: "CE is a systematic approach to integrated product development that emphasizes response to customer expectations and embodies team values of cooperation, trust, and sharing in such a manner that decision making proceeds with large intervals of parallel working by all life-cycle perspectives early in the process, synchronized by comparatively brief exchanges to produce consensus."

Dwivedi and Sobolewski [6] proposed an architecture of a concurent engineering system composed of four levels as follows:

- An object-oriented data base.
- An intelligent data base engine.
- A high-level interface.
- A high-level tools.

In this architecture, several technologies are used to build the system such as:

- object-oriented programming.

- expert systems.
- visual programming.
- database and information retrieval.

To integrate the work among different teams and sites working in a big projects, there must be some kind of synchronization to facilitate the communication and co-operations between them. A concurrent engineering infrastructure that encompasses multiple sites and subsystems called Pallo Alto Collaborative Testbed (PACT), was proposed in [3]. The issues discussed in this work were:

- Cooperative development of interfaces, protocols, and architecture.
- Sharing of knowledge among heterogeneous systems.
- 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 is discussed in [9]. The argument in this work is that object-oriented distributed computing is a natural step forward from the client-server systems of today. A least-common-denominator approach to object orientation as a key strategy for flexibly coordinating and integrating networked information processing resources is also discussed. An automated, flexible and intelligent manufacturing based on object-oriented design and analysis techniques is discussed in [8], and a system for design, process planning and inspection is presented.

Several important themes in concurrent software engineering are examined in [4]. Some of these themes are:

Tools: Specific tool that support concurrent software engineering.

Concepts: Tool-independent concepts are required to support concurrent software engineering.

Life cycle: Increase the concurrency of the various phases in the software life cycle.

Integration: Combining concepts and tools to form an integrated software engineering task.

Sharing: Defining multiple levels of sharing is necessary.

A management system for the generation and control of documentation flow throughout a whole manufacturing process is presented in [5]. The method of quality assurance is used to develop this system which 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 address the coordination and communication problems in engineering, was developed at the MIT Intelligent Engineering Systems Laboratory [11]. In their project they address several research issues such as, frameworks, representation, organization, design methods, visualization techniques, interfaces, and communication protocols.

Some important topics in software engineering can be found in [7], such as, the lifetime of a software system, Analysis and design, module interfaces and implementation, and system testing and verification.

4 The Interface Layer

The prototyping environment for robot manipulators consists of several sub-systems such as:

- Design.
- Simulation.
- Control.
- Monitoring.
- Hardware selection.
- CAD/CAM modeling.

- Part Ordering.
- Physical assembly and testing.

Figure 2 Shows a schematic view of the prototyping environment with its sub-systems and the interface.

There is a lot of shared parameters and information among these sub-systems. To maintain the integrity and consistency of the whole system, a multi-site interface is proposed with the required rules and protocols for passing information through the whole system. This interface will be the layer between the robot prototype and the sub-systems, and also it will serve as a communication channel between the different sub-systems.

The tasks of this interface will include:

- Building relations between the parameters of the system, so that any change in any of the parameters will automatically perform a set of modifications to the related parameters on the same system, and to the corresponding parameters in the other sub-systems.
- Maintaining a set of rules that governs the design and modeling of the robot.
- Handling the communication between the sub-systems using a specified protocol for each system.
- Identifying the data format needed for each sub-system.
- Maintaining comments fields associated with some of the sub-system to keep track of the design reasoning and decisions.

The difficulty of building such interface arises from the fact that it deals with different systems, each has its own architecture, knowledge base, and reasoning mechanisms. In order to make these systems cooperate to maintain the consistency of the whole system, we have to understand the nature of the reasoning strategy for each sub-system, and the best way of transforming the information to and from each of them.

There are several mechanisms used in these sub-systems which can be classified as follows:

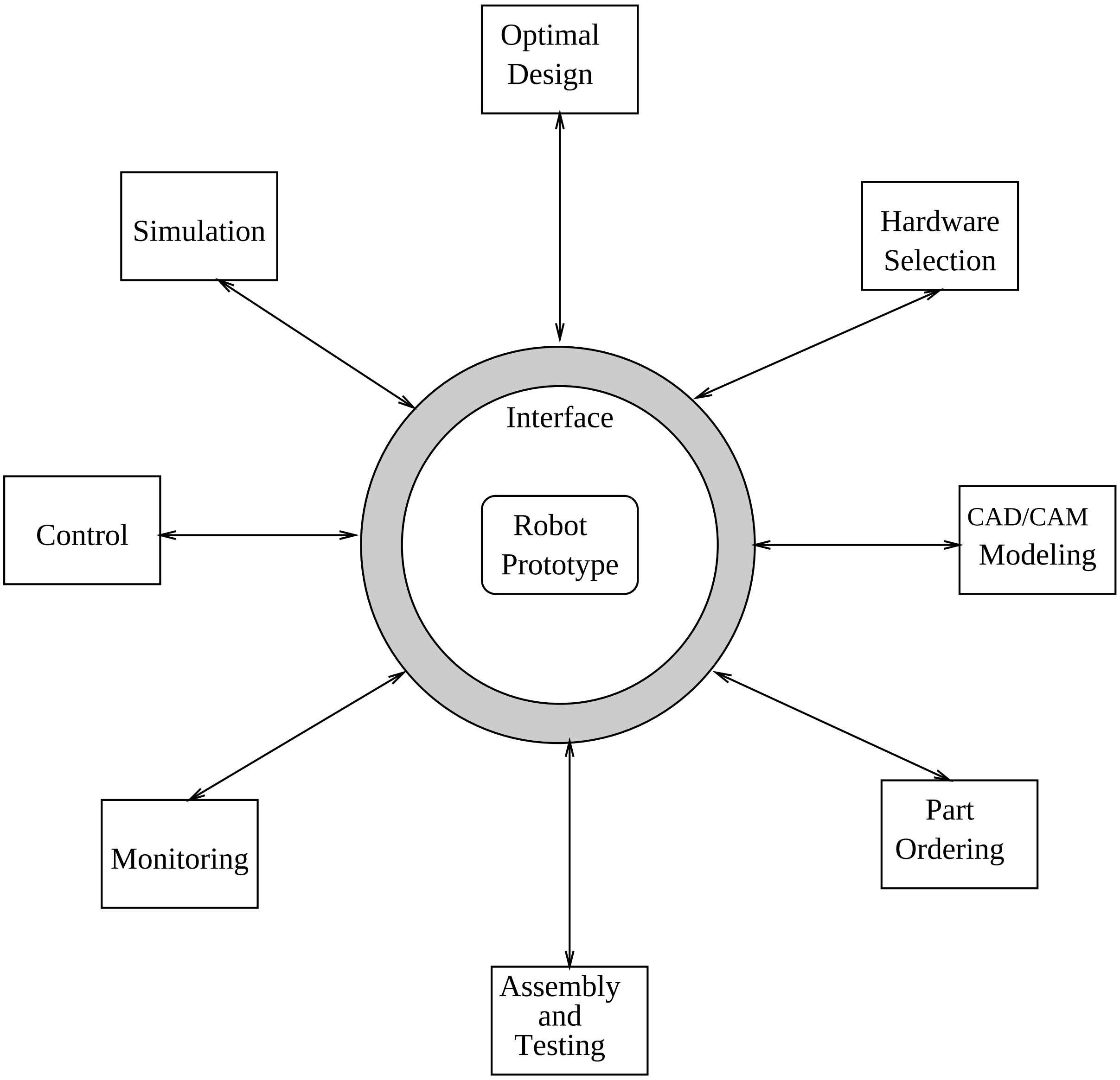

Figure 2: Schematic View for the Robot Prototyping Environment.

- **Constrained-based approach:** this approach is used in the optimal design sub-system.
- **Ruled-based approach:** used in the the CAD/CAM and the hardware selection sub-systems. These rules are used to assist decision making during the design process.
- **Search-based approach:** used in the part-ordering sub-system, which is basically, catalog search for the required parts (motors, sensors, amplifiers, link-materials, etc). This system will be the front-end of an internet-based cataloging system developed at the Mechanical Engineering Department.
- **Functional relations:** used for building the relations between some of the design parameters. For example, link lengths is one of the parameters that has relations with other parameters such as masses and inertia tensors, and also it takes place in the design, control, and simulation systems. If we change the length of one of the links, we want the corresponding mass and inertia tensor to change with a pre-specified functions that relates the length to each of them. We also want the length in the other sub-systems to change as well according to pre-specified mathematical relations.
- **Mathematical Formulation:** used in the simulation, and control sub-systems to define the robot modules (kinematics, inverse kinematics, dynamics, etc).
- **Shared Database Manipulation:** used in most of the sub-systems. For example, the simulation and control are just retrieving data from the shared database, while the monitor subsystem adds analysis information to the database that will be used as a feedback to the design sub-system. The design sub-system updates the parameters of the system. The CAD/CAM system uses this database to check the validity of the chosen parameters and adds to the database some comments about the design and manufacturing problems that might exist.

Since we are dealing with different architectures and approaches, we will use an *object-oriented* scheme to design this interface. Each object deals with one of the sub-systems in its own language. This will make it easier to change the approach or the structure of any of the sub-system without affecting the other sub-systems, by only changing the corresponding object in the interface. Figure 3 shows the proposed interface layer.

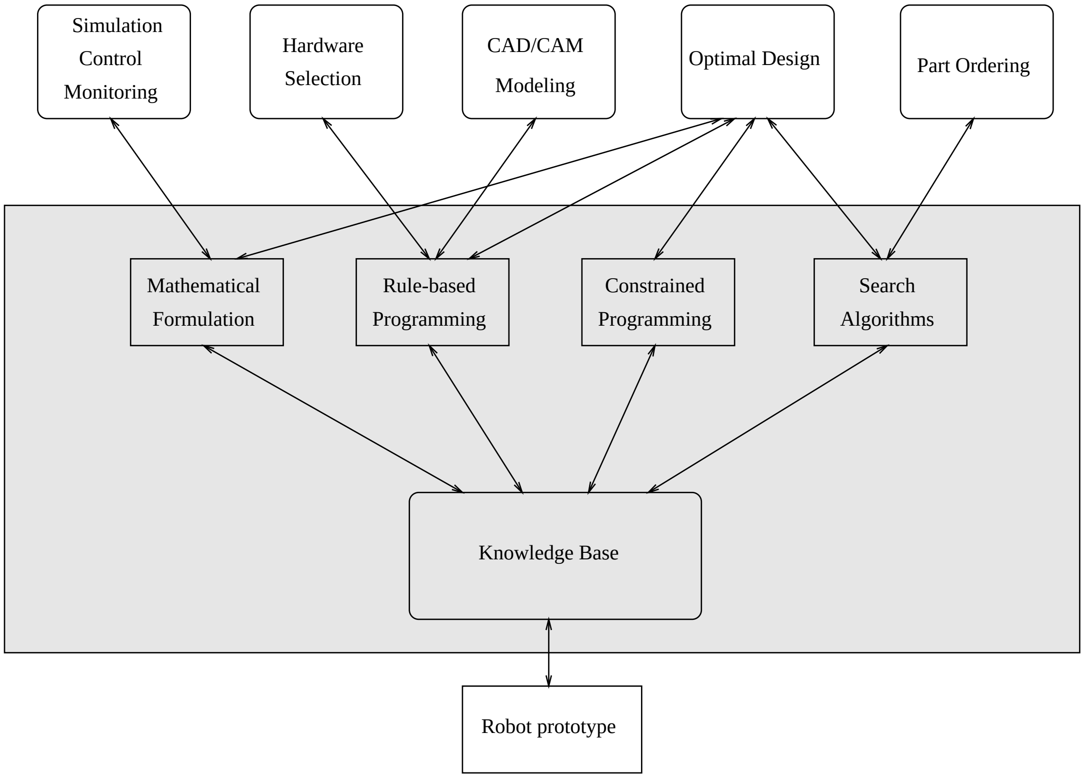

Figure 3: The Interface Between the Subsystem and the Prototype Robot.

In this environment the human role should be specified and a decision should be taken about which systems can be fully automated and which should be interactive with the user. The following example will illustrate the mechanism of this interface and the way these systems can communicate to maintain the system consistency.

Suppose that the designer wants to change the length of one of the links and he wants to see what should be the motor parameters that give the same performance requirements. First, this change will be recorded and the length field will be updated in the shared database for each sub-system. Then the optimal design will be used to determine the new values for the motor parameters using the simulation program. Then search techniques will be used to look up for the motor with the required specifications in the part-ordering system. Here we have two cases: a motor with the required specifications is found in the catalogs, or no motor is available with these specification, in this case, this will be recorded in the comments field and another motor with closest specifications will be selected. Next, The motor specifications will be updated in the database, then the CAD/CAM system will be used to generate the new model and to check the feasibility of the new design. For example, the new motor might have a very high rpm, which requires gears with high reduction ratio. This might not be possible in some cases when the link width is relatively small and a sprocket is hard to install. In this case, this will be recorded in the comments filed and the user will be notified with this problem and will be asked to either change the some of the parameters or the performance requirements and the loop starts again. Once the parameters is determined, the monitoring program will be used to give some performance analysis and compare the results with the required performance, and produces a report with the results.

As another example, suppose that we need to select link masses and motor parameters that give maximum speed and minimum position error. The design sub-system will select density of the links material from the finite density set, and will use the part-ordering system to select the motor and try to get the best combination of motor parameters and link masses that give best value for the combined objective function (speed and position error). The optimization problem here will be solved using the simulation programs. After selecting the required parameters, the CAD/CAM system will be used to generate the model and again to see if this is a valid model to be manufactured. In some cases the motor might be too heavy relative to the link weights (usually when we have small links). In this case the simulation results will show that and another motor should be selected, or another density should be chosen.

Finally the parts can be ordered and the assembly can take place.

4.1 Interaction Between Sub-systems

To be able to specify the protocols and data transformation between the sub-systems in the environment, the types of actions and dependencies among these sub-systems should be identified, also the knowledge representation used in each sub-system should be determined.

The following are the different types of actions that can occurs in the environment:

- Apply relations between parameters.
- Satisfy rules.
- Satisfy constraints.
- Make decisions. (usually the user Makes the decisions).
- Search in tables or catalogs.
- Update data files.
- Deliver reports (text, graphs, tables, etc.).

There are several data representations and sources such as:

- Input from the user.
- Data files.
- Text files (documentation, reports, messages).
- Geometric representations (Alpha-1).
- Mathematical Formula.
- Graphs.
- Catalogs and tables.
- Rules and constraints.

- Programs written in different languages (C, C++, Lisp, Prolog, etc.).

Some of the sub-systems can change some of the parameters and the configuration of the prototype system. The Optimal design sub-system is the one that make almost all the changes in the design parameters. The CAD/CAM sub-system can also make some design changes according to some geometric and manufacturing rules and constraints. The user can change any of the design parameters, make decisions, and run any of the sub-systems.

Tables 1, 2, 3, and 4 describe the interaction between the sub-systems; that is, what each sub-system needs to know when if some of the design parameters are changed by one of the sub-systems or by the user, and what actions it might take as a consequence of this change.

The following is a description for the actions that may take place in the environment as a result of changing some of the design parameters.

Change constraints and optimize: When any change occurs to one of the design parameters, that means changing in one of the constraints for the optimization problem, then the user can decide whither to rerun the optimal design system or not.

Update file: Updating the data files used by the simulation, control, and monitoring sub-systems.

Apply relation: Some of the parameters are related to other parameters in the same sub-system, and to corresponding parameters in other sub-systems. For example, the relation between the link length in the design sub-system and the corresponding drawing length in the monitoring sub-system can be something like:

$$L_{Monitor} = LinkScale * L_{Design},$$

where, $linkScale$ is the scaling factor to draw the link on the computer's screen.

Select D/A, D/A chips: When the motors and the sensors are selected, a chip that contains the D/A and A/D converters and the micro-programs that control the conversion should be selected by the hardware selection sub-system.

Select platform: According to the selected update frequency and the number of computation in each sub-system, the hardware selection sub-system will select the machines that can accommodate that frequency.

System	*Lengths*	*Masses*	*Motors*	*Frequency*	*Sensors*	*Feedback*	*Friction*

Table 1: The interaction between the user and the sub-systems.

System	*Lengths*	*Masses*	*Motors*	*Frequency*	*Sensors*
Simulation	Update file	Update file	Update file	Update file	Update file
Control	Update file	Update file	Update file	Update file	Update file
Monitoring	Apply relation Drawing length		Apply relation Max torque	Apply relation Display rate	Apply relation Max pos,vel
HW Selection			D/A chip	Select platform	A/D chip
CAD/CAM	Update model Check for length	Update model Check for weight	Update model Check gear ratio		
Part-ordering		Search and give report			Search and give report
Assembly	Change link Test, report	Change link Test, report	Change motor Test, report		Change sensors Test, report

Table 2: The interaction between the optimal design and the other sub-systems.

System	*Lengths*	*Masses*	*Gear Ratio (Motor)*
Simulation	Update file	Update file	Update file
Control	Update file	Update file	Update file
Monitoring	Apply relation Drawing length	Update file	Apply relation Max torque
HW Selection			Select D/A chip
Optimal Design	Optimize for other parameters	Optimize for other parameters	Optimize for other parameters
Part-ordering			Search and give report
Assembly	Change link	Change link	Change motor and gears

Table 3: The interaction between CAD/CAM and the other sub-systems.

System	*Platform (Update rate)*	*Communication (Feedback rate)*
Simulation	Update file	Update file
Control	Update file	Update file
Monitoring	Apply relation Display rate	Apply relation Display rate
Optimal Design	Optimize for other parameters	Optimize for other parameters

Table 4: The interaction between hardware and other sub-systems.

Update model: The CAD/CAM sub-system will create a new model for the prototype robot according to the changes in the design parameters.

Check for length, mass, gear ratio and friction: Apply the rules and constraints for each of these parameters that are imposed by geometric and manufacturing limitations.

Search and give report: After the motor specification and the sensor ranges are selected, the part-ordering sub-system will search in the parts catalogs to find the required motors and sensors. If no motors or sensors are found with the required specifications, this will be reported to the user, and the some other motors or sensors with close specifications will be recommended.

Change parts, test, and report: This is the final step in the design. After all parameters are selected and all parts are available, the assembly process can take place, then the design can be physically tested, and the results are reported to the user.

In some cases there might be interaction cycles. in such cases, the user has to take decisions that resolve these cycles. For example, suppose that the link length was determine by the design sub-system, but the CAD/CAM system has some rules that

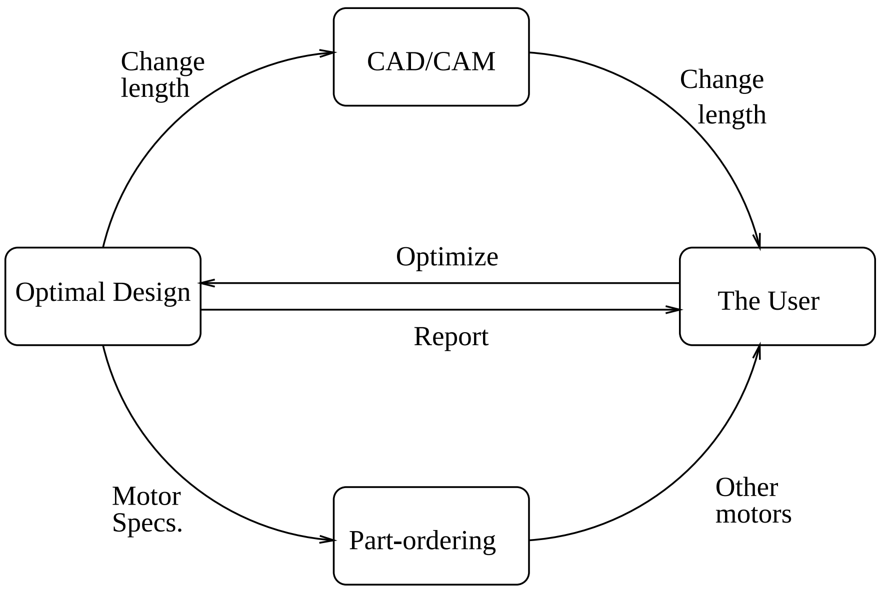

Figure 4: Examples of Some Interaction Cycles.

requires the length to be changed. In this case the design sub-system needs to be run again to accommodate this change. this might change the motor parameters or sensor ranges, and again, this change may violate another rule in the CAD/CAM sub-system which requires another change, and so on. To resolve this cycle the user can take some design decisions that will satisfy the rules and constrains in the sub-systems.

The part-ordering sub-system can cause some indirect changes to the design parameters. For example if a motor with certain specifications is not found, then this sub-system will report that to the user and may recommend some other motors that have close specifications to the required. The user then can either choose one of the recommended motors, or make some design changes and run the optimal design sub-system to get new motor specifications. Figure 4 shows some interaction cycles.

4.2 The Interface Scheme

There are several schemes that can be used for the interface layer. One possible scheme is that: each sub-system will have a sub-system interface (SSI) which has the following tasks:

- Transfer data to and from the sub-system.

- Send requests from the sub-system to the other interfaces.
- Receive requests from other sub-system interfaces and translate it to the local language.

These sub-system interfaces can communicate in three different ways, (see Figure 5):

Direct connection: which means that all interfaces can talk to each other. the advantage of this method is that it has a high communication speed, but it makes the design of such interfaces more difficult, and the addition of new interface or changing one of the interfaces requires the modification of all other interfaces.

Message routing: in this scheme, any request or change in the data will generate a message on a common bus and each SSI is responsible to pick the relevant messages and translate it to its sub-system. The problem with this scheme that it makes the synchronization between the sub-systems very difficult, and the design of the interface will be more complicated.

Centralized control: in which all interfaces will talk with one centralized interface that controls the data and control flow in the environment. The advantages of this scheme is that it makes it much easier to synchronize between the sub-systems, and the addition or modification of any of the SSIs will not affect the other SSIs. The disadvantage is that it has lower communication rates than the other two methods.

5 Object Analysis

The interface layer contains several components that define the objects in the environment, the relation between these objects, the rules and constraints in the system, the representation of these objects in each sub-system, and the communication protocols between the sub-systems.

Object analysis approach will be used to determine the system components and functions, and the relation between them. The following is a description of the system objects.

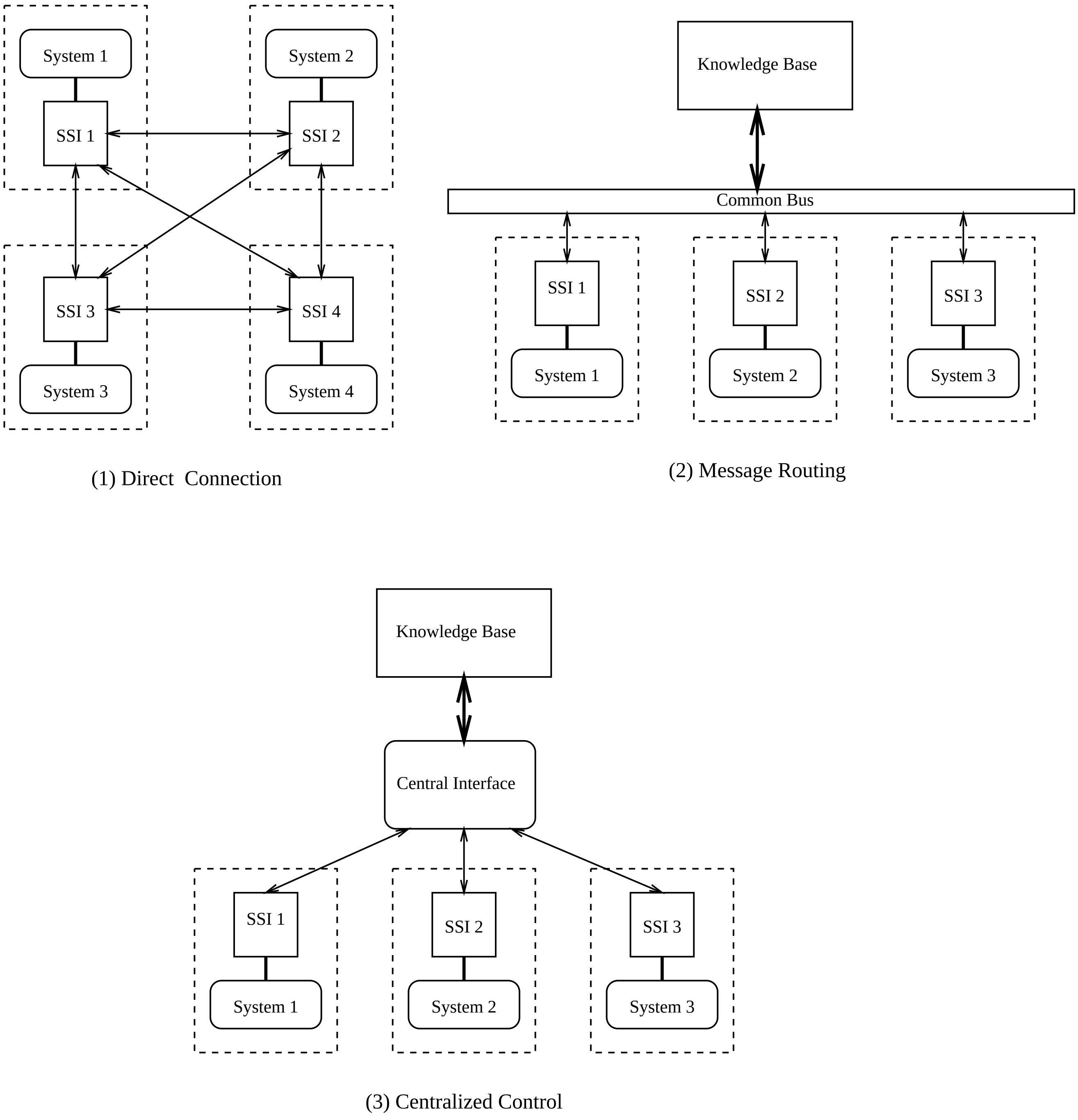

Figure 5: Three Different Ways for sub-system interfaces communication.

- Robot-prototyping
  - Robot
  - Rules
  - Constraints
  - Relations
  - communication
  - Sub-systems
  - Reports
  - Actions
  - Hardware-setup
  - Performance-measures
- Robot
  - Robot-configuration
  - Control
  - Input
  - Results
- Rules
  - Parameters
  - Sub-systems
  - Description
- Constraints
  - Parameters
  - Sub-systems
  - Description
- Relations
  - Object-fields

  - Sub-systems
  - Relation-type
  - Relation-formula

- Communication
  - Protocols
  - Messages
  - Routing

- Sub-systems
  - Optimal-design
  - Simulation
  - Control
  - Monitoring
  - CAD/CAM
  - Part-ordering
  - HW-selection
  - Assembly-and-testing

- Reports
  - Report-type
  - Source
  - Destination
  - Report-contents

- Actions
  - Action-type
  - Sub-system
  - Action-parameters
  - Report

- Hardware-setup
  - Platform
  - Wiring
  - A/D-D/A-converters
- Performance-measures
  - Position-error
  - Velocity
  - Power-consumption
  - Manipulability
  - Structured-length-index
- Robot-configuration
  - Degrees-of-freedom
  - Links
  - Joints
  - Offset
  - Motors
  - Sensors
- Control
  - Torque
  - Voltage
  - Update-rate
  - Sensor-rate
  - Feedback-gains
- Input
  - Input-type
  - Time-period

- Desired-Trajectory

- Results
  - Actual-position
  - Simulated-position
  - Actual-velocity
  - Simulated-velocity

- Links
  - Length
  - Density
  - Inertia-tensor
  - Cross-section

- Joints
  - Type
  - Friction

6 Conclusion

A flexible prototyping environment for electro-mechanical systems in general, and for robot manipulators in particular is proposed. So far we have implemented some of the sub-systems such as: controller, simulator, and monitor, We are now in the stage of testing the three-link robot, implementing the optimal design sub-system, and putting the basis for the shared knowledge base and the interface layer.

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] Cleetus, K. J. Difinition of concurrent engineering. *CERC Technical Report Series* (1992).

[3] 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.

[4] Dewan, P., and Riedl, J. Toward computer-supported concurrent software engineering. *IEEE Computer Magazine* (Jan. 1993), 17–27.

[5] Duhovnik, J., Tavcar, J., and Koporec, J. Project manager with quality assurance. *Computer-Aided Design 25*, 5 (May 1993), 311–319.

[6] Dwivedi, S. N., and Sobolewski, M. Concurrent engineering - an introduction. *CERC Technical Report Series* (1990).

[7] Lamb, D. A. *Software Engineering; Planning for Change.* Prentice Hall, 1988.

[8] 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.

[9] Nicol, J. R., Wilkes, C. T., and Manola, F. A. Object orientation in heterogeneous distributed computing systems. *IEEE Computer Magazine* (June 1993), 57–67.

[10] Sobh, T. M., Dekhil, M., and Henderson, T. C. Prototyping a robot manipulator and controller. Tech. Rep. UUCS-93-013, Univ. of Utah, June 1993.

[11] Sriram, D., and Logcher, R. The MIT dice project. *IEEE Computer Magazine* (Jan. 1993), 64–71.