VII. Conclusion

In this paper, optimal path planning for robot manipulator is presented that utilizes a variational technique. A simplified robot arm is considered to facilitate geometric collision checking scheme. This is incorporated in the variational technique to develop a collision avoidance strategy. The collision checking scheme is rather stringent. However, it reduces the computational requirements.

Different path planning problems are considered in an unified treatment. The method adopted is very flexible and can handle various criterion functions. It can also take into account different kinds of obstacles. The efficacy of the method is demonstrated through numerical examples and digital simulation of a PUMA 560 type of robot arm. The obstacles considered in the numerical examples are simplistic in nature. This however, is not a restriction as the method can be extended to consider multiple static or dynamic obstacles at an increased cost of computation.

Finally, a couple of words about the scope of this paper. Firstly, the path planning procedures proposed here are very effective in repetitive pick-and-place operations rather than in assembly operations. In a much cluttered workspace, the objective is to choose a path that avoids all the obstacles and to travel cautiously through it. Optimum paths are fairly meaningless in such situations. Secondly, the global convergence property of the MLV have not been proved. It is possible that such variational technique may lead to a local optimum. However, this should not cause any difficulty to the MTPP problem. In the MEPP problem the final trajectory may depend on the nominal trajectory chosen. One way of choosing the nominal trajectory is using the joint interpolation technique. One can also use the trial and error method. In either case, the trajectory chosen will not be very far away from the optimal path and the MLV can be effectively applied. In cases where there are many posisible choices of the nominal trajectory, it is advisable to compute the optimal path for each and compare them.

References

[1] M. E. Kahn and B. Roth, "Near minimum time control of open loop articulated kinematic chains," *Trans. ASME J. Dyn. Syst., Measurement, Contr.*, vol. 93, pp. 164–171, 1971.

[2] J. Y. S. Luh and W. M. Walker, "Minimum time along the path for a mechanical arm," in *Proc. 16th Conf. Decision and Contr.*, vol. 1, New Orleans, LA, 1977, pp. 755–759.

[3] J. Y. S. Luh and C. S. Lin, "Optimum path planning for mechanical manipulators," *Trans. ASME J. Dyn. Syst., Measurement, Contr.*, vol. 102, pp. 142–151, 1981.

[4] A. Ghosh and P. Balamuraleedhar, "Near-minimum-time trajectory planning in Cartesian space," in *Proc. 4th Int. Conf. on CAD, CAM, Robotics and Factories of Future*, New Delhi, India, 1989.

[5] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," *Trans. ASME J. Dyn. Syst., Measurement, Contr.*, vol. 106, pp. 102–106, 1984.

[6] J. F. Bobrow, S. Dubowsky and J. Gibson, "Time optimal control of robotic manipulators along specified paths," *Int. J. Robotic Res.*, vol. 4, no. 3, pp. 3–17, 1985.

[7] J. F. Bobrow, "Optimal robot path planning using the minimum-time criterion," *IEEE J. Robotics Automat.*, vol. RA-4, pp. 443–450, 1988.

[8] K. G. Shin and N. D. McKay, "Minimum time control of robotic manipulator with geometric path constraints," *IEEE Trans. Automatic Control*, vol. AC-30, pp. 531–541, 1985.

[9] ——, "Selection of near minimum time geometric paths for robotic manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 501–511, 1986.

[10] ——, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 491–500, 1986.

[11] A. Ghosh and A. M. Patrikar, "Optimum path planning using the method of local variations," in *Proc. SME World Conf. Robotic Res.*, Maryland, 1989. (Also *SME Trans. Robotics Automat.*, 1990.)

[12] L. A. Krylov and F. L. Chernous'ko, "Solutions of problems of optimal control by the method of local variations," *U.S.S.R. Comput. Maths. & Math. Physics*, vol. 6, pp. 12–31, 1966.

[13] A. Ghosh, C. Seshadri and A. M. Patrikar, "Time optimal path planning for robot manipulators," in *Proc. IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, 1990.

[14] T. Lozano Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 681–698, 1981.

[15] ——, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robotics Automat.*, vol. RA-3, pp. 224–238, 1987.

[16] R. A. Brooks, "Planning collision free motions for pick and place operations," *Int. J. Robotic Res.*, vol. 2, no. 4, pp. 19–44, 1983.

[17] E. G. Gilbert and D. W. Johnson, "Distance functions and their applications to robot path planning in presence of obstacles," *IEEE J. Robotics Automat.*, vol. RA-1, pp. 21–30, 1985.

[18] S. H. Suh and K. G. Shin, "A variational dynamic programming approach to trajectory planning with a distance safety criterion," *IEEE J. Robotics Automat.*, vol. RA-4, pp. 334–349, 1988.

[19] J. W. Boyce, "Interference detection among solids and surfaces," *Comm. ACM*, vol. 22, pp. 3–9, 1979.

[20] B. H. Lee and C. S. G. Lee, "Collision-free motion planning of two robots," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, pp. 21–32, 1987.

[21] C Seshadri and A. Ghosh, "Minimum-time trajectory planning for two robots," *Proc. IEEE Ind. Electron. Soc. Conf.*, CA, 1990.

[22] C. Seshadri and A. Ghosh, "Optimum path planning for two robots—A variational approach," *Proc. Int. Symp. Intell. Robotics*, Bangalore, India, 1991.

[23] C. P. Neuman and V. D. Tourassis, "Discrete dynamic robot models," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 193–204, 1985.

A Hypertext Software Package to Help Document System Designs

William L. Chapman and A. Terry Bahill

Abstract— **A hypertext-based software package was created to help engineers, with little formal systems engineering experience, to document a system design. A hypertext-based object-oriented programming environment was used to allow easy transfer of information within the design and to provide a user friendly interface. Tradeoff analysis was automated. This allowed fast sensitivity analysis within the tradeoff studies and greatly decreased the time needed to complete the concept exploration. Graduate students extensively tested the software. Ease in writing the documentation and completing the analysis was a major benefit. Drawbacks were the software's slow speed and its inability to automatically reenter extracted data.**

I. Introduction

Documenting a system design is a mundane but important task. Most systems are documented poorly if at all. This causes a breakdown in communicating the requirements to the designer and results in a nonoptimized system design. We built a hypertext-based object-

Manuscript received September 19, 1990; revised August 14, 1992.

W. L. Chapman is with the Hughes Aircraft Company, P.O. Box 11338, Tucson, AZ 85734.

A. T. Bahill is with the Department of Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85721.

IEEE Log Number 9206199.

oriented tool to help document system designs. This software package is called the systems engineering design software (SEDSO).

In this paper, we will first explain the elements of system design that must be documented. Next, there will be a description of an example of system design documentation. We then describe our software tool SEDSO that allows easy creation of the systems engineering documentation. This software guides the systems engineer through the documentation of the design effort. When completed, all the requirements necessary for conceptual development and evaluation will have been rigorously defined and the tradeoff and sensitivity analysis completed. Finally we examine the use of the software by students to formally document and analyze their class project designs.

II. Model-Based Systems Engineering

Model-based systems engineering [1], [2], which is similar, but not identical to, the systems engineering procedures used in most large companies, divides the systems requirements into six categories.

1. Input/output and functional.
2. Technology.
3. Input/output performance.
4. Utilization of resources.
5. Tradeoff.
6. System test.

The input/output and functional requirement defines the time scale, inputs, outputs, and functions the system must perform. It represents what the system must do independent of the technology.

The technology requirement defines what the system can be built with and typically consists of the components available and their individual characteristics such as cost, availability, schedule, reliability, etc., for the system design.

The input/output performance requirement is used to measure how well the input/output and functional requirement is met. This is typically done through figures of merit and performance indices.

The utilization of resources requirement is used to measure how well the technology requirement is met. This is also done through the use of figures of merit and performance indices such as project cost, schedule, environmental impact, etc.

The tradeoff requirement is used to decide the tradeoffs between the input/output performance and utilization of resources requirements that must be made to choose the best system.

The system test requirement is used to describe how the system requirements will be evaluated and figures of merit measured on a real system. In addition, the criteria for observance, compliance, conformance and acceptance are given.

To describe the system design, seven systems engineering documents are used.

1. Problem situation document.
2. Operational need document.
3. System requirements document.
4. System requirements validation document.
5. Concept exploration document.
6. System functional analysis document.
7. System physical synthesis document.

The problem situation document is the executive summary. It explains the problem that needs to be solved and states who the customer and designers are. It is written in plain language and is intended for management.

The operational need document is used to define, in plain language, what the customer expects from the new system. The needs of the system are described by using the six categories of requirements mentioned above. This document is intended for management, the customer and systems engineers.

The system requirements document is used to mathematically, or in complete textual detail, describe each of the requirements addressed in the operational need document. Its audience is systems engineering.

In the system requirements validation document we examine the mathematical description of the input/output requirements presented in document 3 to check for consistency, demonstrate that a real world solution can be built, and show that a real world solution can be tested to prove that it satisfies the input/output and functional requirements. Often identifying an existing system that satisfies all the requirements is enough to complete the system validation. This document is written for systems engineering.

The concept exploration document is used to develop several different concepts. This is done through the use of tradeoff studies and modeling to determine which of the concepts is superior. This document will be rewritten many times as more information becomes available. It is written for systems engineering.

The system functional analysis document is used to decompose the chosen concept into successively smaller functions that will eventually be simple enough to implement physically. Its intended audiences is systems engineering.

The system physical synthesis document is used to break the system functions from document 6 into successively smaller physical units until the design is complete. It is created in conjunction with system functional analysis.

SEDSO was built to simplify the creation of these documents.

III. Sierra

A project from an undergraduate microcomputer class was chosen to test the software package and provide a small engineering project for which we could create a complete set of systems engineering documents. The students had to create a controller for two trains that run on two intersecting circular tracks. The trains can collide at the intersections. The controller must interface to existing location detectors and power controllers to prevent collisions. The requirements were provided to the students and they then built the systems and documented the results. This project is known as the Systems & Industrial Engineering RailRoad Assignment or SIERRA. The students created three different controllers: one with integrated circuits, another with an assembly language program for a Motorola 68000 microcomputer, and the last with a Pascal program. This project has been done by more than 500 students since 1986. The old student reports were the control group that we compared to the engineering reports generated with SEDSO.

The full documentation of this system design as produced with SEDSO is contained in [8, ch. 6]. This was the first full implementation, available to the general public, of the seven systems engineering documents outlined in model-based systems engineering [1]. The documentation for SIERRA was created as an example of a good systems engineering documentation report. SIERRA is 82 pages including exhibits.

IV. Systems Engineering Design Software

It was decided that a software package that could be used to create the documentation of the system design would help engineers design systems. This software would create the seven systems engineering documents used to track the requirements development and concept exploration phases of the life cycle. This new software is called the systems engineering design software (SEDSO).

After investigating several commercial software packages a product called HyperPAD [10] was selected for building SEDSO. This product is an object-oriented programming language for IBM compatible

Section 2.4 - ENTER THE I/O PERFORMANCE REQUIREMENTS

Figure of Merit [1] Name [Number of Collisions]

Description

The number of times the trains come into physical contact.

Relative Importance [8]

Fig. 1. A performance requirement template.

TABLE I
TYPICAL WEIGHTS FOR THE FIGURES OF MERIT

Figure of Merit	Importance Value 1 to 10	Weight IWi
1) Number of Collisions	8	0.258
2) Trips by train A	7	0.225
3) Trips for train B	7	0.225
4) Spurious stops by A	3	0.096
5) Spurious stops by B	3	0.096
6) Availability	2	0.064
7) Reliability	1	0.032

personal computers. The package has a strong programming language coupled with an easy to use interface.

SEDSO begins a new system design by prompting the systems engineer to enter requirements information. All the information for each requirement explained above is used. SEDSO documents the system design by using the requirement information to fill in the blanks in the hypertext fields for each systems engineering document. The questions help engineers provide all the requirements needed in the project. Data was entered only once then used in several different documents. For example, the user entered data for performance and resource requirements once in document 2 and the data was subsequently used in documents 3 and 5 for evaluating the system.

Templates for the design approach were provided. For example, the I/O requirements have a field for the system states. The "boiler plate" symbology for the modeling language was already filled in with the terminology for the system and the student needed only to fill in the blanks. After the systems engineer has completed the state diagram this data is easily entered. To enter a new performance requirement a template with edit fields is provided as shown in Fig. 1.

Automated creation of the weights for each figure of merit is accomplished by having the systems engineer enter the data into a template similar to that shown. The relative importance of each figure of merit based on a scale of 1 to 10 is entered. These numbers are summed and a weighted value between 0 and 1 is assigned by SEDSO as the normalized weight (IWi). Table I shows an example of weightings used on SIERRA.

Sublevels of figures of merit can also be entered. These sublevels are computed separately and the overall value passed up to the main level. This allows the breakdown of large figures of merit into manageable pieces. We seldom used more than seven figures of merit at any level.

The scoring and tradeoff analysis was also automated. We used scoring functions to scale different figures of merit to values between 0 and 1. Calculation of the standard scoring functions are done based on values for upper, lower, baseline, and slope parameters entered. Instead of 12 scoring functions of Wymore [1], SEDSO defines only one scoring function that encompasses all 12. The four basic shapes that can be derived from our scoring function are shown in Fig. 2.

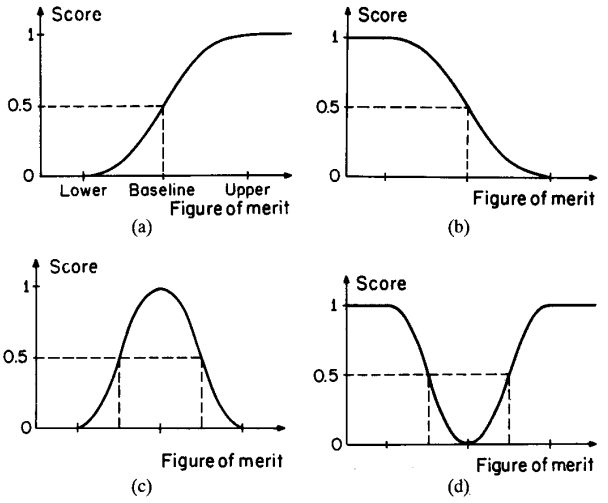

Fig. 2. The four shapes that our scoring function can assume.

Equations for these scoring functions are given in [8]. The software required the systems engineer to enter values of infinity for upper thresholds when there was no upper limit or negative infinity for lower thresholds when there was no lower limit. For example in SIERRA the scoring function shown in Fig. 2(a) was created to evaluate the number of trips for the model trains. In this case a lower limit of 0 was set and an upper limit of infinity. The baseline or expected value of FigureMerit was 0.5. The baseline parameter indicates what figure of merit yields a score of 0.5 on a scale from 0 to 1. The slope is measured at the baseline and showed how quickly the score changed at that point. The standard scoring function accepted the value of the observed number of trips and returned a scaled score between 0 and 1. For example, for 8 trips a score of 0.917 was returned. For 10 trips a score of 0.982 was returned. These functions are versatile and can be used in other domains such as for activation functions in artificial neural networks and for membership functions in fuzzy logic knowledge-based systems.

The tradeoff was computed automatically when all the data had been entered. Overall scores for each concept were computed allowing a comparison of how well each conceptual design met the requirements. Although in decision analysis many different means are available for calculating tradeoffs we used a linear tradeoff between performance requirements and resource requirements. In SIERRA both were weighted equally so a value of 0.5 was assigned to TW1 and TW2. The following equation was used for the tradeoff analysis:

$$TF0 = IF0 * TW1 + UF0 * TW2$$

where $IF0$ is the overall input/output performance index, and $UF0$ is the overall utilization of resources index.

This automation of the decision theory made it much easier for the systems engineer to compute the tradeoffs and to do the sensitivity analysis needed in the concept exploration in document 5. After changing the weights, figures of merit, or parameters of the scoring function, new tradeoff scores were automatically computed. These were compared with previous scores to determine how sensitive the design was to small changes in the data. For example, Table II shows approximate values, or blue sky guesses, for the number of trips completed by each train.

With these estimates the overall I/O Performance Index is 0.963. We then played a "what-if game" and asked what would happen if

TABLE II
APPROXIMATE VALUES FOR I/O FIGURES OF MERIT

Requirements	FigureMerit	Score	IWi
1) Number of Collisions	0	1	0.258
2) Trips by train A	8	0.917	0.225
3) Trips by train B	8	0.917	0.225
4) Spurious stops by A	0	1	0.096
5) Spurious stops by B	0	1	0.096
6) Availability	1	1	0.064
7) Reliability	1	1	0.032
Overall Performance Index = 0.963			

TABLE III
REVISED APPROXIMATE VALUES FOR I/O FIGURES OF MERIT

Requirements	FigureMerit	Score	IWi
1) Number of Collisions	0	1	0.258
2) Trips by train A	9	0.961	0.225
3) Trips by train B	10	0.982	0.225
4) Spurious stops by A	0	1	0.096
5) Spurious stops by B	0	1	0.096
6) Availability	1	1	0.064
7) Reliability	1	1	0.032
Overall Performance Index = 0.987			

we changed the approximated values for the number of trips for Train A and Train B as shown in Table III.

The increased number of trips resulted in a higher scaled score, which when multiplied by the appropriate weight and summed gave a new value of 0.987. In other words a design that allowed each train to make more trips had an higher overall score. This score was the basis for computing the tradeoff and determining the best system. To conduct a sensitivity analysis the user repetitively changed a value for a figure of merit and the software computed the new overall scores. Likewise, the weights or the scoring function could be changed for a figure of merit in documents 2 or 3 and a new value computed in document 5. The time required to change a weight, scoring function or figure of merit and obtain a new overall score was only a few minutes. A new document with the updated information was then immediately available. A more rigorous method for sensitivity analyses has been given in [11].

After the user entered all the requirements data for a document, a copy was printed or sent to a disk file. This was accomplished by SEDSO by inserting the requirements information into the hypertext fields. The fields were embedded in standard specification text. The output allowed checking of the information by others involved in the project and provided a permanent record of the system design. Each printed copy carried a date stamp for control purposes.

Hypertext is a means of interlinking data that is not tied to specific fields as it is in most flat file database schemes [3], [4]. This feature was a helpful tool in this project since data was used in several separate documents and cross checking was useful without locking users into a fixed framework. It also allowed searching all fields in the document for a phrase. This was needed when updating the documents. Other system theories such as quality function deployment (QFD) [5], [6] can also be implemented using this hypertext approach.

The alternative that gets the highest score in the tradeoff analysis of document 5 is called the "recommended alternative" and its value is assigned to a variable that is printed in the output text paragraph. Often when we present the recommended alternative, our customers say "This is not the right answer." So we open up document 3 and ask which of the weights or parameters they want to change. We make the changes and record the source of the changes. Then SEDSO automatically recomputes the tradeoff study, and presents a new recommended alternative. This process continues until the customers are happy. At the end of this process an alternative has been recommended, the reasons for its superiority are made manifest, and the sources of the design decisions have been documented. This documentation is important for subsequent operation and replacement of the system.

SEDSO required a total software development time of about 180 hours with an additional 60 h of tests. This number is low because of the use of the object-oriented tool, however the size of the program is large at about 552 Kbytes. This is because of the integration of the database with the code. Because of the speed in creating the software, the size of the final program was not an issue.

V. STUDENT TESTING

Students in a graduate systems engineering course were asked to document a system design for a class project. The students were given the requirements for conducting a Cub Scout Pinewood Derby race. Students divided themselves into four groups comprised of 3 to 5 students and were given the option of generating the documentation with or without SEDSO. Three of the groups chose to use SEDSO and one did not.

The students had one month to complete their projects, which averaged 100 pages in length and took on average 200 total student hours. With the help of SEDSO as a software tool and SIERRA as an example, these students were able to create the complete set of system engineering documentation in a reasonable period of time. The team that decided not to use the SEDSO package to document their system design delivered a project report that was not internally consistent. Different writing styles were obvious and all requirements were not documented with the same detail level. However, their report was much better than similar reports done in previous years: this was attributed to the SIERRA example. The instructor felt that the documents that used SEDSO were much more consistent and had better modeling and analysis. Two of the groups generated the results with SEDSO then reformatted it using a word processor. Their documentation had the best appearance.

However, the students who used SEDSO felt that it was too slow and did not have enough word processing features (such as a spelling checker) that they were accustomed to from personal computers. In addition bugs in the SEDSO software resulted in loss of data on several occasions. Generally the aspect of the software the students liked the most was the automatic computation of the scoring functions and the consistent transfer of requirements information from one document to the next. We felt that the students had a much better understanding of systems theory and documentation practices from using the software and completing the projects. The results indicate that SEDSO was of help to the students, but that a more complete and trouble free software package is needed. The amount of quality documentation generated in a short time by a team of students was clearly linked to the SEDSO software's ability to accurately track the requirements and output the results.

It is appropriate for students in a graduate level Systems Engineering course to use SEDSO to produce all seven system engineering documents for a particular system. SEDSO and SIERRA help. In addition we have found that undergraduate students in our Microcomputer Systems and Expert Systems courses also benefit from writing these documents. But time constraints limit use to requesting only documents 1, 2, 3, and 5. SEDSO and the Pinewood Derby

Documentation [8, ch. 5] provide excellent demonstration and lecture material for these courses. SEDSO is being used every semester at the University of Arizona.

VI. Discussion

By writing our own system design documentation software we learned a considerable amount about what should be in a system design report. Many of the features we put into SEDSO were not considered before the design began. For example, we did not initially appreciate how long the documents would be and their storage requirements. Also the more user friendly the software became, the better the student's documentation also became. Because we used a hypertext environment, we decided to prompt the user to enter weights, scoring functions, test methods, test results and sensitivity analysis for each requirement stated by the customer. This changed the way we thought of the reports, particularly during the concept exploration phase. We never "lost" a requirement or failed to address it, because a blank paragraph would have been output in our report. We also encouraged the designers to include four types of data in the concept exploration document: the Present System (if one exists), Approximation, Simulation and, when appropriate, a Prototype.

Developing SEDSO dramatized these four iterations of the concept exploration document. Many proposed systems are similar enough to an existing system so that the existing system can be used to provide initial values for the figures of merit. For SIERRA we had hundreds of previous student projects that could be used. In the design of a mass rapid transit system for a metropolitan area, the existing system of streets and roads can be used for figures of merit. However, for other proposed systems no similar system might exist, such as a manned station on Mars.

All systems should have an iteration based on approximations, or educated guesses, made by experts. Most systems can have an iteration based on a simulation of the system. Finally, many systems will have prototypes that can be used to generate figures of merit. In order to be useful the prototypes would have to be low in cost compared to the final system, otherwise it would not be economical to throw away prototypes that are not chosen. The field of expert systems is epitomized by fast low-cost prototypes. It is the rule, rather than the exception, to throw away the first prototype. This allows several prototypes to be made for each system. These prototypes can be used to provide figures of merit for the tradeoff studies. This process of evaluating each concept numerous ways would not have been put into every set of documentation if it were not automated.

We made a mistake designing SEDSO: we did not include spelling checkers and text processors. So the users took SEDSO's output and processed it with their spelling checkers and text processors. Unfortunately, SEDSO could not accept this processed output as its input. Therefore, the users had to also reenter the changes with SEDSO. We could have ameliorated this problem by using a hypertext tool that used ASCII files so that the users could correct spelling, grammar and formatting on the original files. However, at the time no such tool existed. If we were to rewrite SEDSO we would use a newer hypertext tool that could do this, or perhaps C++.

RDD-100® is a commercial systems engineering software package that addresses this problem. It has one database with a dozen facets. One person can use a graphics editor on a block diagram; this does not merely change the graphics display, but instead it changes the fundamental database. Another person can edit a text file. Once again this does not simply change the text file, it changes the underlying database. A third person could alter the data with a spreadsheet, once again they would not be simply changing the values in the cells, but they would be altering the database.

Our system was implemented using a 12-MHz IBM AT environment but the students created their reports on a 5-MHz IBM XT. This proved too slow for their use, especially when outputting copies of the entire set of documents. More modern workstations would obviously improve the performance of the system.

Other requirement tracking schemes appear in the literature. Automated design systems can be used for keeping a library of existing technologies and pruning the set for a viable solution [7], [12]. Commercial packages, such as Ascent Logic's RDD-100, provide standard system modeling output, such as IDEF charts and flow diagrams, and can be used to track some requirement information [9]. We have found no package that will simultaneously track the requirements, aid in concept selection via a tradeoff study and output system requirement documentation besides SEDSO.

VII. Conclusion

Systems engineering methodology is of great practical use. The most useful aspect we found was the documentation of the entire system design. This project has helped by creating the first complete set of seven systems engineering documents, as defined in Model-based Systems Engineering [1], available to the public. The software package SEDSO was instrumental in creating the documentation for SIERRA and for automating the decision theory used in concept selection.

A more complete software package would be of benefit to the students. SEDSO was the foundation upon which a complete system can be built. Students who used SEDSO created more complete documentation in less time than students who did not use SEDSO. In addition requirements were carefully tracked throughout the design and concept selection tradeoffs were automated enabling a thorough sensitivity analysis. A package with stronger word processing tools and better response is also desirable.

The example SIERRA was of particular importance to the students. By having a simple design to follow from document 1 through document 7 the students could easily see the benefits of system design. It is felt that with SIERRA and SEDSO, system design techniques are more easily understood and are more useful for students.

Acknowledgment

We thank Dr. Wayne Wymore for spending time reviewing SIERRA and SEDSO and ensuring its correctness and accuracy. His expertise in systems design was of great practical use.

References

[1] A. W. Wymore, *Model-based Systems Engineering.* Boca Raton, FL: CRC Press, 1993.
[2] A. W. Wymore, *System Engineering Methodology for Interdisciplinary Teams.* New York: Wiley, 1976.
[3] J. Conklin, "Hypertext: An introduction and survey," *Computer,* vol. 20, no. 9, pp. 17–41, 1987.
[4] J. H. Walker, "Supporting document development with Concordia," *Computer,* vol. 21, no. 1, pp. 48–59, 1988.
[5] B. King, *Better Designs in Half the Time.* Methuen, MA: GOAL/QPC, 1989.
[6] Y. Akao, *Quality Function Deployment: Integrating Customer Requirements into Product Design.* Cambridge, MA: Productivity Press, 1990.
[7] J. W. Rozenblit and Y. M. Huang, "Rule-based generation of model structures in multifaceted modeling and system design," *ORSA J. Computing,* vol. 3, pp. 330–344, 1991.
[8] W. L. Chapman, A. T. Bahill, and A. W. Wymore, *Engineering Modeling and Design.* Boca Raton: CRC Press, 1992.
[9] *RDD-100 User's Manual.* San Jose: Ascent Logic Corp., 1991.
[10] *HyperPAD User's Guide.* New Canaank, CT: Benchmark, 1989.

[11] W. J. Karnavas, LP. Sanchez, and A. T. Bahill, "Sensitivity analyses of continuous and discrete systems in the time and frequency domains," *IEEE Trans. Syst., Man, Cybern.*, pp. 488–501, this issue.
[12] J. Grady, *Systems Requirements Analysis.* New York: McGraw-Hill, 1993.
[13] S. Khoshafian and R. Abnous, *Object Orientation.* New York: Wiley, 1990.

Robust Adaptive Controller Design and Stability Analysis for Flexible-Joint Manipulators

R. A. Al-Ashoor, R. V. Patel, and K. Khorasani

Abstract—**The problem of controlling robot manipulators with flexible joints is considered. A reduced-order flexible-joint model based on a singular perturbation formulation of the manipulator equations of motion is used. The concept of an integral manifold is utilized to construct the dynamics of the slow subsystem. A fast subsystem is constructed to represent the dynamics of the elastic forces at the joints. A composite adaptive control scheme is developed with special attention to stability and robustness of the controller. The proposed controller is based on on-line identification of the manipulator parameters and takes into account the effect of a class of unmodeled dynamics, identification errors and parameter variations. Stability analysis of the resulting closed-loop full-order system is presented. To show the capability of the proposed algorithm, an example of a two-link flexible-joint manipulator is considered. Simulation results are given to illustrate the applicability of the proposed control scheme.**

NOMENCLATURE

Unless mentioned otherwise, the following notation is used in this paper. Matrices are denoted by bold letters.

n	Number of joints.
$\boldsymbol{I}$	$n \times n$ identity matrix.
τ	$n \times 1$ generalized joint torque vector.
$\theta(\dot{\theta}, \ddot{\theta})$	$n \times 1$ joint position (velocity, acceleration) vector.
$\boldsymbol{M}$	$2n \times 2n$ positive-definite inertia matrix.
$\eta(t)$	$n \times 1$ torque vector representing unmodeled dynamics in the slow subsystem.
$e(\dot{e}, \ddot{e})$	$n \times 1$ generalized joint position (velocity, acceleration) error vector.
$\hat{\boldsymbol{\Theta}}$	Matrix of estimated parameters.
Ψ	Vector of measured variables.
ε	Elasticity parameter.
ψ	$n \times 1$ elastic joint force/torque vector.
$\phi(\dot{\phi}, \ddot{\phi})$	$n \times 1$ actuator position (velocity, acceleration) vector.
γ	Integral manifold of ψ.
ξ	Deviation of ψ from the integral manifold (transient behavior of ψ).
$\boldsymbol{J}_l, \boldsymbol{J}_m$	$n \times n$ moments of inertia corresponding to the links and motors, respectively.
$\boldsymbol{B}_l, \boldsymbol{B}_m$	$n \times n$ centrifugal and Coriolis terms for the links and motors, respectively.
k	Stiffness of the flexible joints.
R_l, R_m	$n \times 1$ gravity and friction terms for the links and motors, respectively.

Manuscript received July 1, 1991; revised May 26, 1992.
This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under grants OGP0001345 and OGP0042515.
The authors are with the Department of Electrical and Computer Engineering, Concordia University, Montreal, PQ, H3G 1M8, Canada.
IEEE Log Number 9205807.

I. INTRODUCTION

Joint elasticity in a robot manipulator implies that the position of an actuator (i.e., the angle of the motor shaft) is not directly related to the position of the driven link. From the modeling point of view this internal deflection can be approximated by inserting a linear torsional spring at each joint. As a consequence, the rigid arm dynamic model has to be modified in order to describe completely the relation between applied torque and link motion [1]. Most industrial robots employ DC or AC motors connected in series with harmonic drives (high-torque, high-ratio gear boxes) used mainly for speed reduction. In some applications, transmission belts, or long shafts in the drive system (usually in the joints) are also used. The joint elasticity results in lightly damped oscillatory modes in the open-loop response of the system [2]. To capture the aforementioned behavior, the manipulator is modeled by a chain of rigid sublinks interconnected by elastic joints [10].

It is shown in [3], [4] that the control schemes that assume a rigid model for the manipulator are limited in their applicability to real robots where the assumption of perfect rigidity is never satisfied exactly. The resonant behavior in some range of frequencies imposes bandwidth limitations on any control algorithm that is designed assuming perfect rigidity. This may cause stability problems for feedback control laws that neglect joint elasticity [7]. For quasi-static applications, simplified models that consider only the dynamics of the drive system have been used by Kuntze *et al.* [8]. Spong [10] has investigated a simplified model that neglects the inertial coupling between the actuators and links. Models including full nonlinear dynamic interactions among joint elasticities and inertial properties of links and actuators have been introduced by Nicosia *et al.* [9].

Recently several advanced control algorithms for flexible-joint manipulators have been proposed. Approaches using singular perturbation techniques [11], sliding modes [12], pseudolinearization [13], and model reference adaptive control [14] have been developed. A method based on the concept of integral manifold was suggested by Khorasani *et al.* [5], [6] and Spong *et al.* [7]. In this approach the control algorithm is designed by assuming perfect knowledge of the parameters of the flexible-joint manipulator. De Luca [1] uses dynamic feedback for linearization also with no uncertainties. A theoretical study of robust control was performed by De Wit and Lys [2]. Their approach uses a two-step estimation procedure for the unknown parameters of the manipulator. Stability analysis of the resulting closed-loop system was not investigated when significant parameter variations are permitted in the open-loop system. Since the estimation of the rigid body dynamics depends on the elasticity of the joints, a two-step estimation procedure for flexible-joint manipulators is not possible.

Different adaptive control schemes for rigid manipulators have been proposed in the literature [28], [29] to circumvent the difficulties arising from parametric uncertainty. We can essentially classify these works into two different categories. In the first category, the adaptive