

An Evolution of Collaborative Design Tools

Daniel R. Kuokka
Lockheed Research Labs
Orgn/96-20, Bldg/254F
3251 Hanover Street
Palo Alto, CA 94304
kuokka@aic.lockheed.com

1 Introduction

This paper describes a series of three design tools intended to support various forms of conflict management in concurrent and collaborative engineering settings. The tools are Explorer, ParMan (Parameter Manager), and PCA (Project Coordination Assistant). Explorer is a single-user parametric design assistant that allows many alternative designs satisfying multi-disciplinary constraints to be generated and compared graphically. ParMan is a multi-user parametric design agent that permits distributed engineers to input constraints and evaluate the impact on the combined design space. PCA is a multi-user tool allowing engineers to encode and share semi-structured, non-parametric information about a project. Each of these tools is described below, along with experiences and motivation for development of the subsequent tool.

2 Explorer

Explorer [5] supports intelligent, highly interactive search of concurrent engineering trade spaces under conflicting objectives. Explorer provides a data framework in which engineers from multiple disciplines can specify the basic constraints and metrics governing the design task. Given this initial (usually large) database, Explorer presents an interface by which an individual designer can enter specific constraints and metrics. Upon the user's request, Explorer automatically generates all consistent solutions to a design specification that satisfy the static constraints as well as the specific design requirements. Finally, after all solutions have been enumerated, Explorer provides interactive, graphical utilities that aid the user in determining the best solution. By concentrating on the computation and intuitive presentation of the trade-space, Explorer allows engineers to resolve conflicts stemming from the various requirements placed on the design.

Any class of design problems that can be represented as a set of parameters can be encoded within the framework, where parameters can be real-valued numbers or enumerated values (examples will be drawn from an Explorer application for Printed Circuit Board (PCB)

Construction design). An application is set up as follows: First, a set of parameters is defined, and arbitrary constraints among the parameters are entered in tabular or functional format (input to the system as files). This represents the static constraints for a particular domain (e.g., only certain materials are available and they cost so much).

Once the application is set up, the user brings up the dynamic specification worksheet (Figure 1, bottom left window). This worksheet allows the user to input a set of design goals such as maximum allowable cost, minimum allowable yield, and desired impedance; and specific engineering constraints and preferences, such as availability of certain processes, manufacturing cost factors, and reliability factors.

Once the specification is complete, Explorer generates all designs by computing values for parameters that satisfy the constraints and requirements (top right window). For the PCB example, the designs are in terms of number of layers, design class, board size, etc. The designer can then browse the designs via graphical visualization tools in order to determine the best design (bottom right window). In the figure, the cost, yield, and routability have been graphed, sorted by increasing cost. This graphical presentation allows the relationships among the parameters and metrics to be quickly understood, and often clearly differentiates the good and poor designs.

In experiments with Explorer, the underlying enumeration paradigm provides a convenient facility for the exploration of alternative designs. Most notably, Explorer reveals options that are otherwise not considered, allowing designers to perform more complete trade-off analyses. In addition, by enumerating and presenting all feasible designs, it provides the information needed by engineers to resolve design conflicts rationally. In tests run with PCB design engineers, Explorer invariably found (or made apparent) novel design configurations. Even expert designers, upon using Explorer for a few minutes, expressed surprise at some of the designs that propagated to the top after graphing and sorting. These designs often yielded comparable performance and cost while offering better manufacturability and reliability.

Another discovery is that even though the design spaces are extremely large, typical constraints reduce this space to a magnitude (on the order of 500 for the PCB domain) that makes automatic enumeration feasible (but still well beyond the capability of an unaided human). This is a key finding, since the ability to manipulate, sort, and compare complete designs is vital to the success of the system. Whereas a parametric paradigm does not capture many aspects of design (particularly topology and structure), it has been shown to be useful in many cases. Finally, even though Explorer provides significant automated reasoning, it keeps the human in control of the decision loop, using automated reasoning to facilitate rather than dictate.

Explorer is a powerful trade-off analysis tool, but it assumes a great deal of a priori knowledge about the multi-discipline design space. Engineers with knowledge of all aspects of the trade-off space, and knowledge of Explorer's data formats, must input the broad range of constraints, tables, and metrics required to characterize the domain. In many cases, this is not practical, since Explorer-literate engineers may not be available. Perhaps more importantly, such knowledge is dynamic and only partially defined. Each responsible engineer should be able to enter and modify the constraints dynamically and incrementally via the interface. This gives rise to ParMan, described below.

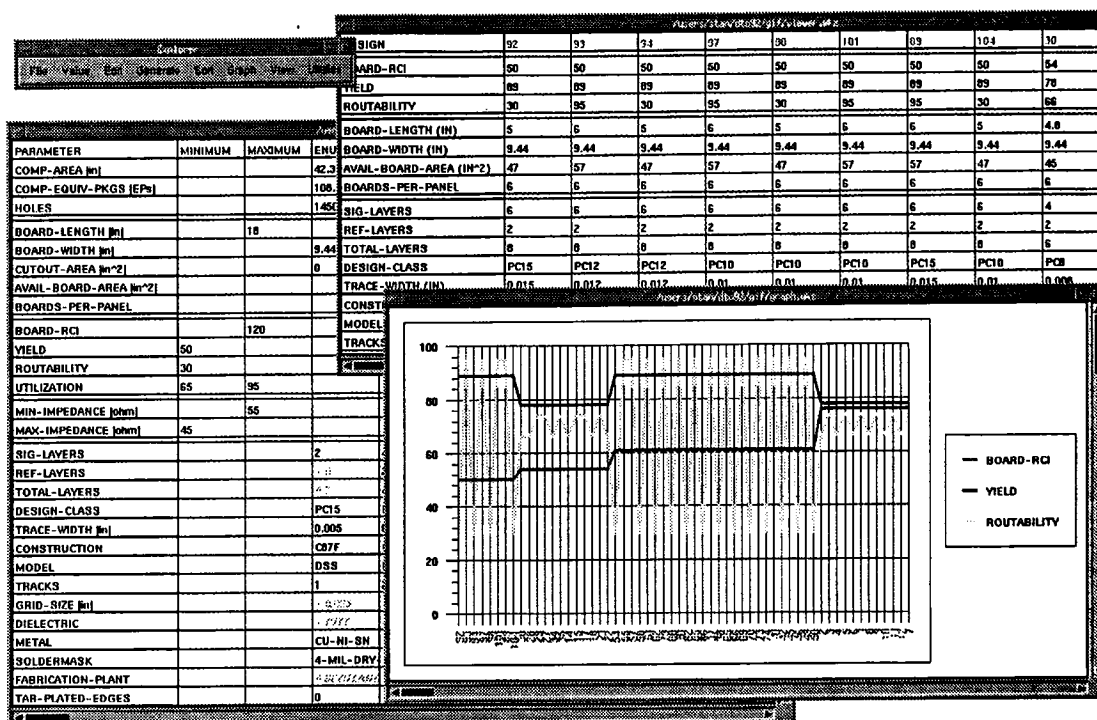


Figure 1: Explorer allows the user to state constraints, generate designs that satisfy those constraints, and view the generated designs.

3 ParMan

ParMan [6] is a collaborative parametric design agent [1, 8, 7] combining the use of agent communication protocols [2], constraint logic programming [3, 4], and a graphical presentation interface. ParMan allows any number of engineers to specify constraints over shared parameters in real time, analyze those constraints locally, export and import constraints to and from other members of the design team, and determine if and how the constraints conflict.

ParMan is used as follows. First, a Project Selector is used to choose from a list of all projects that are of interest to members of the design team. The list is populated dynamically as users create new projects and advertise them to all other agents (ParMan systems and their human users). Once a project is selected, each team member uses a Parameter Graph Editor to define a set of parameters of specific interest. These parameters are organized in a hierarchical graph structure, allowing the user to define components and associate parameters with those components. When a parameter or component is placed in the graph, ParMan advertises it to the other agents. This allows the team to define, dynamically, the design

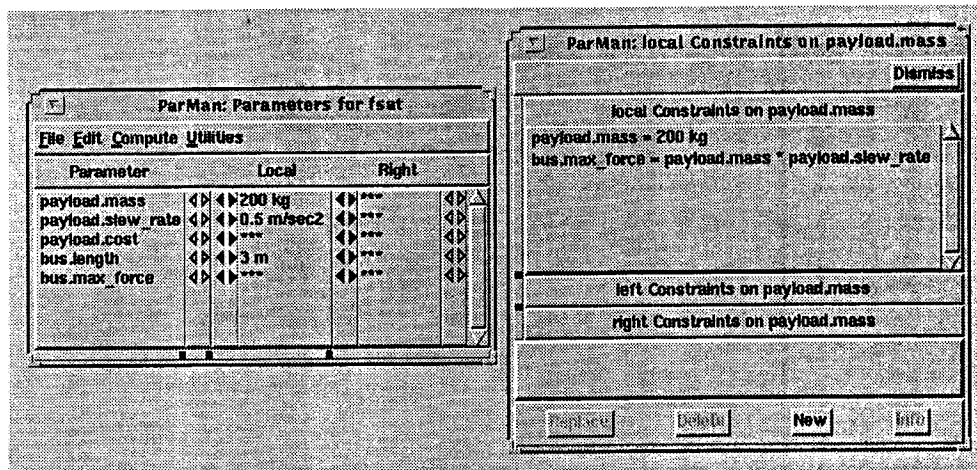


Figure 2: Parameters and simple constraints are displayed on the Parameter Table, more complicated constraints for each parameter are entered and viewed via the Constraint Editor.

space (in fact, this feature of ParMan is a primitive precursor to the Project Coordination Assistant described below).

From the Parameter Graph Editor, each user selects a subset of the parameters to be displayed in a Parameter Table, which is a worksheet of parameters relevant to his aspect of the design, somewhat like Explorer's specification worksheet (see Figure 2, left window). The user can now begin defining constraints over the parameters in the Parameter Table. These constraints are expressed in ParMan's Constraint Language, which includes standard infix arithmetic constraints and an extensible set of predicates. The user can define and analyze constraints locally (the center column), or choose to advertise any or all of the constraints to other agents (by moving them to the right column via the arrows). The user can also ask to be kept informed about any constraints that other agents place on selected parameters (by using the arrows on the far right). Such an action is referred to as subscription, whereas the sharing of constraints is called advertisement.

Given the set of distributed parameters, ParMan provides several tools to aid conflict resolution. The Constraint Grapher presents a graphical display of the parameters and constraints, with links connecting related parameters and constraints (see Figure 3). Constraint nodes are presented in green if the constraint is satisfied. Red is used to indicate constraints that are in direct conflict. Yellow is used to indicate constraints that participate in a conflict, but cannot be directly implicated. The constraint grapher provides means to visualize complex constraint webs, and isolate those constraints contributing to the conflict.

ParMan also includes several other features: a constraint solver, which attempts to find a closed form solution to the set of constraints; a Constraint Tester, which allows the user to drag individual constraints into a test region to isolate problems; a clique finder, which separates the constraints into independent sets; and a units converter, which is used to ensure constraints from multiple sources are expressed in like terms.

ParMan has been used to encode test cases in several domains: the conceptual design of a satellite, a bicycle design domain, and a meeting scheduling domain. In all cases, the

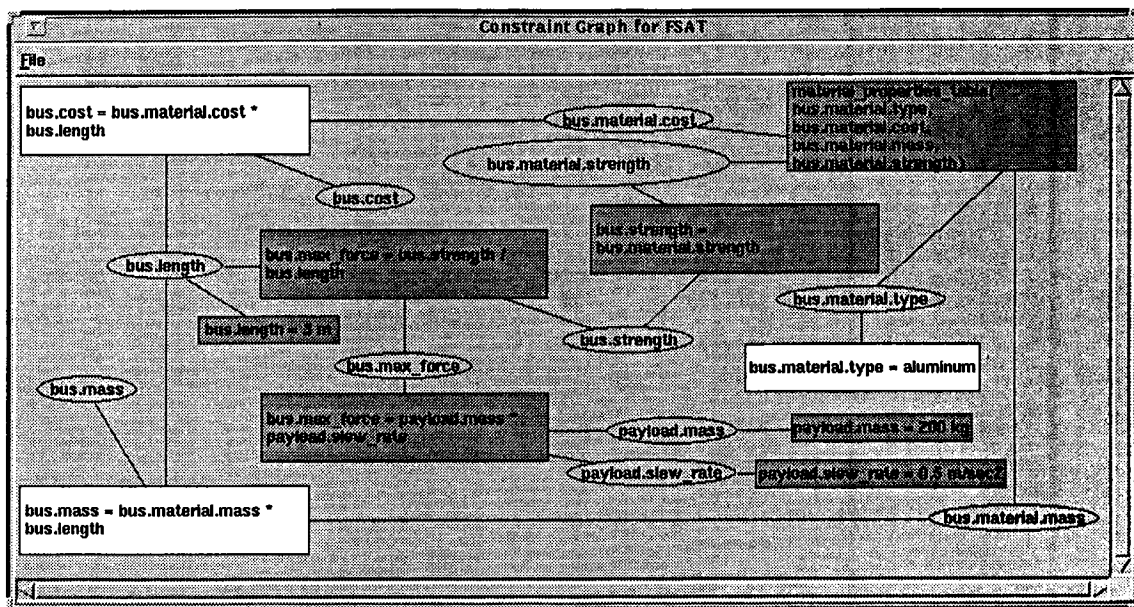


Figure 3: The Constraint Grapher aids in the visualization of relationships among parameters and constraints.

users were geographically distributed (in separate offices). In each test, it is striking how the apparent complexity of the global problem was significantly reduced when distributed among multiple people. Distribution allows each user to focus on and ensure satisfaction of those constraints representing a single perspective.

Once each user assumed a specific role, ParMan proved to be a very natural system for encoding and solving distributed constraint satisfaction problems. Each user's constraints were easily entered, and their appearance as external constraints in other users' interface was natural. Flagging of conflicts by color coding also proved to convey the essential information without complexity. Finally, the facility to find a solution proved quite powerful and beneficial. Unlike Explorer, which only enumerates solutions (albeit quickly and incrementally), ParMan computes a closed-form solution when possible. ParMan also improves upon Explorer by no longer requiring a priori a database of cross-disciplinary design constraints. In ParMan, these are entered dynamically by each specialized engineer. It should be stressed that, however, that Explorer contains many features not implemented in ParMan, so the latter is by no means a superset of the former.

Even with these advances, there are still open issues with ParMan. Like Explorer, the efficiency of the constraint satisfaction algorithms and the expressivity of the language may be insufficient. More relevant to conflict resolution, however, is how the common model assumed by the collaborating engineers is agreed upon. Both systems assume the existence of a well-defined parametric design space. ParMan proposes a primitive solution (the Parameter Grapher) allowing limited specification of the project structure, component hierarchy, and parameters. In any large project, however, where the number and complexity of components increases drastically, this interface will likely prove inadequate. The Project Coordination

Assistant, described below, is an ongoing attempt at addressing this problem.

4 Project Coordination Assistant

Many collaborative design tools, including Explorer and ParMan, are predicated on the existence of a common, shared product and engineering model (the parameters). Even though the creation of such models is being actively pursued as part of efforts like PDES/STEP, the state-of-the-art is still far from sufficient. Also, given the rapid advance of technology, no engineering model can ever be complete and up-to-date, and externally imposed processes are seldom appropriate or accepted. Therefore, there is a need for mechanisms whereby engineers can create and modify a shared model (or schema) of project information dynamically. This permits engineers to define collaboratively the parameter space they must explore.

Once such a capability exists, however, the impact far transcends parametric design tools such as Explorer and ParMan. Amazingly, most problems encountered in large engineering projects revolve around inadequate communication of even the most basic information, such as requirements changes and interface specifications. This leads to improperly specified and integrated components, often resulting in huge project delays and over-runs. In many cases, simply keeping the affected engineers informed of decisions would solve many problems. In essence, engineers need a tool that allows them to avoid the unnecessary conflicts of poor communications as much as they need a tool to resolve constraint-driven conflicts.

The product Coordination Assistant (PCA) allows engineers to organize and view project data in a heterogeneous, semi-structured form. At the simplest level, PCA is similar to the hierarchical component and parameter grapher of ParMan. It allows engineers to define conceptual objects, which serve as containers for and pointers to engineering data. For example, a common class of objects is components, whose pointers, or links, represent the subcomponent relation. Other organizations may be specified, such as a version history or requirements hierarchy. PCA allows engineers to define their own product data model dynamically and collaboratively, rather than relying on an outside data management organization.

PCA is a collaborative tool. Engineers access and modify a shared workspace, allowing each participant to input information right next to, or even in place of, the inputs of the other engineers (issues related to protection, transactions, and access control have not yet been addressed). Thus, comments, extensions, and issues related to others' inputs are naturally linked to their referent. In essence, the system is like a distributed shared whiteboard for semi-structured data, allowing a distributed set of engineers to reify key project information normally communicated in an ad hoc fashion, either verbally or via hard copy.

The basic object structure of PCA is augmented with a number of more advanced capabilities. First, engineers can annotate each object with arbitrary information: text, formulae, requirements, and graphics. Second, engineers can create links to other heterogeneous engineering data, making the basic structure a hypermedia (actually a hyperdata) web. Third, engineers can define rules that monitor for specific events or changes. The rules are specified largely by example, so the engineers do not have to "program" a formal rule base. Thus, engineers can seamlessly browse from requirements to specifications to CAD files to comments, and as changes are made, engineers are automatically notified of changes that affect them.

A view of an initial PCA prototype is shown in Figure 4. PCA is built on top of the World

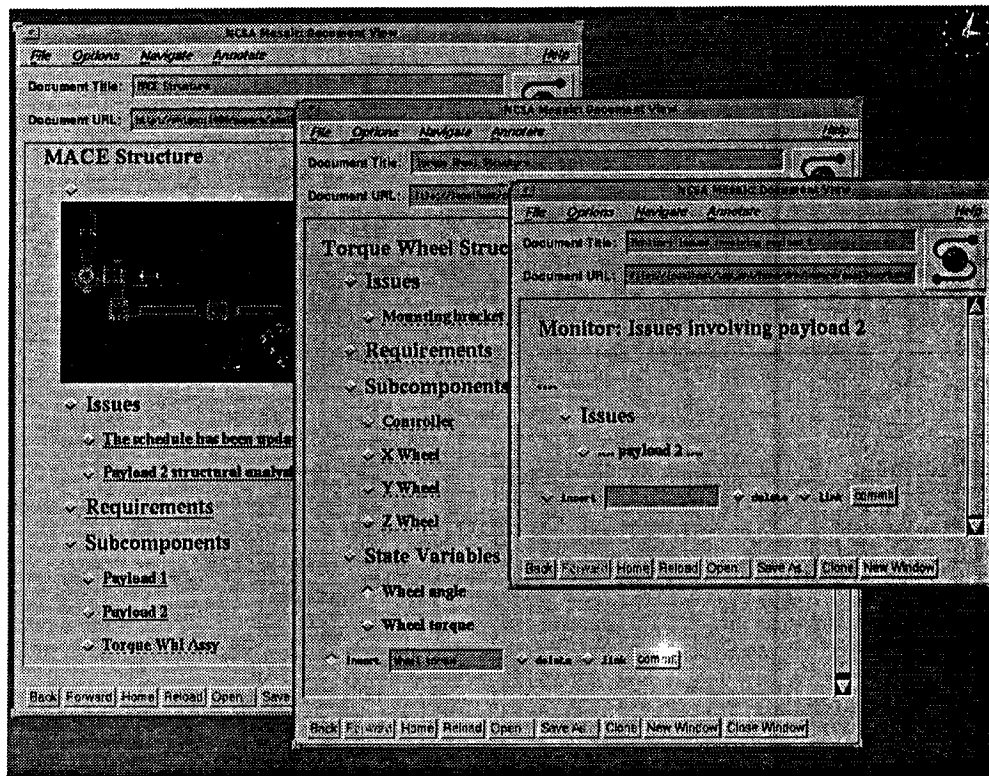


Figure 4: PCA allows multiple users to construct, view, and modify a backbone of engineering information with pointers to heterogeneous data sources.

Wide Web paradigm, making use of HTTP/HTML and Mosaic. This permits heterogeneous information to be accessed, permits information to be contributed by many sources, and ensures cross-platform implementation. The backbone of PCA data is the set of semi-structured objects that many users can edit collaboratively. Two such objects are shown in the figure, one for the MACE satellite, and one for the torque wheel subcomponent, which was accessed by selecting the torque wheel item in the parent.

The underlying rule mechanism can match against each of these forms, with more advanced processing available on the more formal information (i.e., formulae and logical assertions as opposed to text). Thus, the users can set up monitors to alert them if specific wording in a textual specification segment changes (as in the monitor form shown), or if the formal specifications change such that they violate some conditions. In this manner, PCA goes beyond current Product Data Managers since it supports notification over the contents of engineering data rather than notification over the existence or change date of the data.

Since PCA uses a hypermedia paradigm, users can create links to arbitrary data. The example in the figure shows a picture included directly in the object. Not easily shown are the numerous links to other other kinds of data. There is a link to the current project schedule (presumably created via a scheduling application) and a link to the latest structural analysis. The underlying software allows these specific applications to be started up when the user requests to view that data, or a summary of the data created by these complex

applications may be shown as a bitmap.

The development of PCA is still in its early phases, not far enough along to provide significant results. However, the problem being addressed is very compelling, since development of PCA has been in direct response to the needs of several large projects with geographically distributed engineers. This experience has revealed that, even when tools are available, engineers are stymied by the inability to access needed data, or even determine the existence of the data they need. PCA lets the engineers, themselves, define the project data structure rather than waiting for a CAD support group or standards body. The capability to define monitors over this data provides a partial solution to the ever-present problem of propagating changes to affected parties. PCA's support for heterogeneous data is particularly important since the majority of engineering information is not formal CAD data. Engineers are flooded with information like memos, schedules, personnel issues, and requirements, and it is this information that usually slips through the cracks, often having a much greater impact on project performance than technical details. PCA gives engineers a means to organize such information, share it with other team members, and request notification when it changes.

5 Conclusions

Explorer, ParMan, and PCA represent an interesting exploration of the space of collaborative engineering tools. Each includes a set of features that address different aspects of conflict management and resolution. Explorer illustrates a means by which a single engineer can consider a broader set of designs, and evaluate them with respect to criteria and constraints from different disciplines. In this way, conflicts can be detected early in the design phase. On the down side, Explorer requires that information about each of the disciplines be encoded within the tool, posing a significant maintenance and startup cost.

ParMan overcomes the maintenance concern of Explorer by creating a distributed tool. This permits each engineer to input constraints important to him or her, and monitor constraints and decisions made by other engineers. Like Explorer, ParMan includes computation support for identifying conflicts. Whereas ParMan addresses the collaborative nature of engineering better than Explorer, it does not supply the analysis tools that allow engineers to sort, graph, and evaluate many designs.

Both Explorer and ParMan provide significant aid to engineers that know about each other, and that understand the mutual design space. Whereas this is beneficial in an idealized design scenario, few engineering projects are so idealized. Most projects spend much effort simply communicating semi-structured information like component definitions, requirements, specifications, parameters, and interfaces. Furthermore, such definitions are rarely static, which presents the task of communicating changes to the proper engineer. PCA is designed to give engineers a shared persistent workspace that can encompass the heterogeneous data of engineering projects. By the use of PCA, it is hoped that many unnecessary conflicts (those caused by mis-communication) can be avoided.

There is great synergy possible among the relatively unstructured organization capabilities of PCA, the structured visualization of Explorer, and the structured collaboration of ParMan. As elements of a project become well enough defined via PCA, more formal constraints can be defined and monitored via ParMan. When decisions are required, the

Explorer interface can be used to compare and select the best design. The resulting unified tool would provide the means by which engineers can both avoid and resolve the conflicts that are ever present in engineering projects.

6 Acknowledgments

The Explorer system described herein was developed jointly by the author, Stan Jefferson, Lee Barford, and Felix Frayman at Hewlett Packard Laboratories. Parman was developed jointly by the author and Brian Livezey.

References

- [1] M. Cutkosky, R. Engelmores, R. Fikes, T. Gruber, M. Genesereth, W. Mark, J. Tenenbaum, and J. Weber. PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, 26(1), 1993.
- [2] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, J. McGuire, D. McKay, S. Shapiro, R. Pelavin, and C. Beck. Specification of the KQML agent communication language. Official document of the DARPA Knowledge Sharing Initiative's External Interfaces Working Group. Technical Report 92-04, Enterprise Integration Technologies, Inc., 1992.
- [3] J. Jaffar and J. Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*. Association for Computing Machinery, 1987.
- [4] J. Jaffar, S. Michaylov, P. Stuckey, and R. Yap. The CLP(R) language and system. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 14(3), 1992.
- [5] D. Kuokka, S. Jefferson, L. Barford, and F. Frayman. A parametric design assistant for concurrent engineering. *International Journal of Concurrent Engineering: Research and Applications*, Submitted, 1994.
- [6] D. Kuokka and B. Livezey. A collaborative parametric design agent. In *Proceedings of the National Conference on Artificial Intelligence*. AAAI Press, 1994.
- [7] D. Kuokka, J. McGuire, J. Weber, J. Tenenbaum, T. Gruber, and G. Olsen. SHADE: Knowledge-based technology for the re-engineering problem; annual report. Technical report, Lockheed Artificial Intelligence Center, 1993.
- [8] J. McGuire, D. Kuokka, J. Weber, J. Tenenbaum, T. Gruber, and G. Olsen. SHADE: Technology for knowledge-based collaborative engineering. *Concurrent Engineering: Research and Applications*, 3(1), 1993.