DISCOVERY OF DESIGN METHODOLOGIES

CIRRUS SHAKERI
Knowledge Technologies International, Inc.,
Lexington, MA 02421, USA

DAVID C. BROWN
Computer Science Department,
WPI, Worcester, MA 01609, USA

MOHAMMAD N. NOORI
Mechanical Engineering Department,
WPI, Worcester, MA 01609, USA

Abstract. In this paper we present an AI-based approach for the discovery of design methodologies for multi-disciplinary design situations. The approach is based on simulating the design process using a multi-agent system that mimics the behavior of the design team. The system activates the pieces of design knowledge when they become applicable. The use of knowledge by agents is recorded by tracing the steps that the agents have taken during a design project. Many traces are generated by solving a large number of design projects that differ in their requirements. A set of design methodologies is constructed by using clustering techniques to generalize the traces. These methodologies can be used to guide design teams through design projects.

1. Introduction

This this research concerns the multi-disciplinary design of engineered systems. It extends the concept of analysis-by-simulation to the area of engineering design research. Analyzing the behavior of physical systems in engineering applications by computer simulation using mathematical models has been a powerful tool in engineering, reducing costs and time in comparison to physical prototyping and experimentation.

Here the same concept is applied to the design *process* instead of the design product. A computational model in the form of a knowledge-based multi-agent system is built that simulates the design process. By running the simulation under different conditions, and examining the performance, detailed understanding of the design process is gained (Shakeri, 1998) (Shakeri, Brown & Noori 1998).

As for simulations of physical systems, the computational model of the

design process is a simplified one in which the design activities usually carried out by humans are performed by software agents in a simplified manner. We have developed these ideas using the multi-disciplinary domain of robot arm design (Rivin, 1988).

The current practices of multi-disciplinary design are based on ad-hoc strategies for handling the complexities that multiple points-of-view bring to the design process. These strategies solve the problem of complexity at the expense of giving up the potential advantages of diversity. The common methodologies for multi-disciplinary design are based on compromising between different disciplines rather than collaborating between them. These methodologies do not use a systematic, holistic approach to the problem of multi-disciplinary design and thus they are not as efficient and effective as they could be.

The most common strategy to overcome the complexities of multi-disciplinary design is 'sequential design', in which different disciplines take part in the design process sequentially. In sequential design, information sharing between different disciplines is limited to the interfaces between disciplines (Levitt, Jin & Dym, 1991). As a result, conflicts between disciplines are not discovered until they are very expensive to resolve, because their resolution may need to destroy the partial designs generated by the previous discipline.

"In sequential design, a tentative design synthesis is developed by one designer, often acknowledged as the *lead discipline* designer, which addresses some of the key performance specifications and constraints" (Levitt, Jin & Dym, 1991). Having a lead discipline that makes key decisions reduces the number of conflicts. The other disciplines conform to the decisions made by the lead discipline, but that may prevent them from producing their best solutions. In this approach a single point-of-view dominates the decision making process, favoring constraints from that discipline, producing lower quality designs, and increasing the number of iterations required to reach an answer.

2. Simulation of the Design Process

Our new approach to the problem of producing better design methodologies for multi-disciplinary design is based on the integration of different disciplines. The discipline-sequential approach, while poor, is quite simple. Integration tends to make the design process more complicated. To overcome this complexity, a computer system was developed based on a multi-agent systems paradigm in order to automate the simulation of the design process.

The system simulates examples of multi-disciplinary design processes while applying integration principles to the problem. These include common design knowledge representation schemes; common communication mechanisms; design knowledge sharing among participants; cooperative problem-

solving strategies among participants; simultaneous design processes where possible; and mechanisms for conflict discovery and resolution.

The large chunks of discipline-specific knowledge are broken into small pieces and are represented in the system by agents. Agent activation is triggered in an opportunistic manner and is unaffected by discipline boundaries. Agents might participate sequentially or in parallel.

During the course of the design process, the traces of the agent activations (i.e., knowledge use) are recorded. The recorded traces consist of patterns of different design tasks that have led to the solution. Some candidate design methodologies are extracted by generalizing the patterns using clustering. Some of these candidates will be reinforced by solving more examples and accepted as design methodologies for that particular class of problems.

A design methodology is a scheme for organizing reasoning steps and domain knowledge to construct a solution. It provides both a conceptual framework for organizing design knowledge and a strategy for applying that knowledge (Sobolewski, 1996). A design methodology can provide the knowledge for decomposing the problem into sub-problems, synthesizing partial designs, evaluating and then combining them into more complete partial designs, ordering design tasks by considering proposals from all participants, and discovering and resolving conflicts.

Figure 1 shows one of the methodologies generated by the system for robot design. The phrase “do the design” means to generates values for the design parameters that are closely tied to the value chosen using the methodology. In this example, choosing the safety factor allows the designer to calculate the dimensions of the cross section of the robot’s arm.

IF

- constraints on deflection and the gain are both tight; and
- requirements on workload are rather “easy”;
- workspace is of type “small-M”;

THEN

- **choose** the location of the base of the robot: “left of or below the workspace length”
- **choose** the material: “steel stainless AISI 302 annealed”
- **select** the shape of the cross section of the link: “hollow round”
- **choose** the structural safety factor: “3”
    - **do** the design and proceed to the next step
- **choose** the link 2 to link 1 length ratio: “0.5”
    - **do** the design and proceed to the next step
- **pick** the configuration of the arm: “left-handed”
- **select** the ratio of section dimension to min. required: “4”—if it fails select “3”
    - **do** the design and proceed to the next step
- **find** the accessible region: use Equation 2-4
- **find** the deflection of the tip: use Equation 2-14
- **choose** the type of controller: “PD”
    - **do** the design and finish the process.

Figure 1. A Methodology Discovered

3. Related Research

Recently there has been increasing recognition that multi-disciplinary design is important. A large amount of very good research has been focused on Multi-disciplinary Design Optimization (MDO) (Sobieszczanski-Sobieski & Haftka, 1996). MDO tries to produce an effective product by using appropriate combinations of parameters to be controlled and optimized by the designer. A key part of the process is the use of decompositions (Sobieszczanski-Sobieski & Haftka, 1996) (Gebala & Eppinger, 1991) (Liu & Brown, 1994).

In multi-disciplinary design problems the values of design parameters may determine what design method will be employed, as methods may have applicability conditions. As different design methods may introduce different dependencies, dependency chains, and problem decompositions, can be dynamically determined. Hence the sequencing of design tasks can also be dynamically determined. Some approaches provide user interaction to help determine task sequences (Kroo & Takai, 1990) (Hale et al., 1996) (Wujek et al., 1996). However, while Multi-disciplinary Design problems often require

the user's investigation of design trade-offs, for each problem and related set of requirements, there are a number of common design task sequences that are used. Such sequences form the basis of a design methodology for that problem or class of problems.

Our work requires that the results of the discovery process be well integrated and concurrent. Fine-grained tasks are needed, as opposed to the large grained tasks used by some research (Hale et al., 1996) (Woyak, Malone & Myklebust, 1995). Our agent-based approach can accommodate qualitative, experiential, and heuristic knowledge. Lander (1997) provides a detailed review, while other work on multi-agent Systems in Concurrent Engineering is reported in (Brown, Lander & Petrie, 1996).

The use of Machine Learning methods in support of design has been well documented (Duffy, 1997). While the use of Case-Based Reasoning (CBR) and inductively formed user (i.e., designer) models is becoming familiar, our method for generalizing design traces is not. Depending on what is included in the design traces, and its representation, we can take advantage of work on clustering, induced finite-state transition networks, inductive learning for state-space search, or flexible macro-operators (Langley, 1996).

4. The Multi-Agent Design System

A knowledge-based model of design is adopted in order to implement the proposed integration strategies. A knowledge-based design tool based on a multi-agent architecture was developed that simulates the design process. The multi-agent paradigm intuitively captures the concept of deep, modular expertise that is at the heart of knowledge-based design (Lander, 1997).

An agent is a self-contained problem solving system capable of autonomous, reactive, pro-active, social behavior. It is a powerful abstraction tool for managing the complexity of software systems (Wooldridge, 1997). A multi-agent system is composed of multiple interacting agents, where each agent is a coarse-grained computational system. Agents are used as an abstraction tool for conceptualizing, designing, and implementing the knowledge-based design approach.

A Java-based computer program called RD (Robot Designer) has been implemented for parametric design of a two degrees of freedom (2-DOF) planar robot arm. The architecture of the system is shown in Figure 2. There are three different layers in the system: *Data*, *Control*, and *Flow.*

The data layer contains the design requirements and design constraints defined by the user at the beginning of each design project. The data layer also contains the state of the design process at any moment and the description of the product as it evolves during the process. Database agents update data and

answer the queries of the other agents. A coordinator agent manages the consistency of the data between different database agents and synchronizes the updates and queries.

Figure 2 also shows how different agents are responsible for gathering, storing, and providing different types of shared knowledge. These agents are `DesignState`, `DesignRequirements`, `DesignProduct`, `Tracer`, `DesignConstraints`, and finally `DatabaseCoordinator`, responsible for gathering data and distributing it among the aforementioned agents.

The control layer contains the design knowledge as well as the knowledge for how to use the design knowledge. In Figure 2 each `Designer` *m_n* agent is responsible for a specific design method *n* in discipline *m* (`k` is for kinematics, `s` for structural, and `c` for control design of a robot arm).

The rest of the agents in the control layer are responsible for coordination and carrying out generic design tasks such as evaluation of the partial designs. They discover and provide the dependency between designers, and provide an agenda for various design tasks such as backtracking.

The flow layer of the system contains a mechanism for communication among agents based on sending and receiving messages. This mechanism consists of a registry and a message passing protocol. Each message has its own thread for processing, that not only provides concurrency between agents, but also it allows each agent to handle multiple messages simultaneously.

Agent activation is triggered in an opportunistic manner. As a result, the dependency between designer agents is discovered on-the-fly when the design process is moving ahead. Figure 3 shows an example of a dependency graph that is dynamically generated during the design process.

The design process starts with a set of designer agents that can use the design requirements and generate a set of values for their output parameters. This set of designers form the first row of the dependency graph with Depth 0 in the graph. Based on the input-output dependency between the design methods a new set of designer agents step forward and generate values for their output parameters. As a result of this process new rows are added to the graph at new depths until the design is complete.

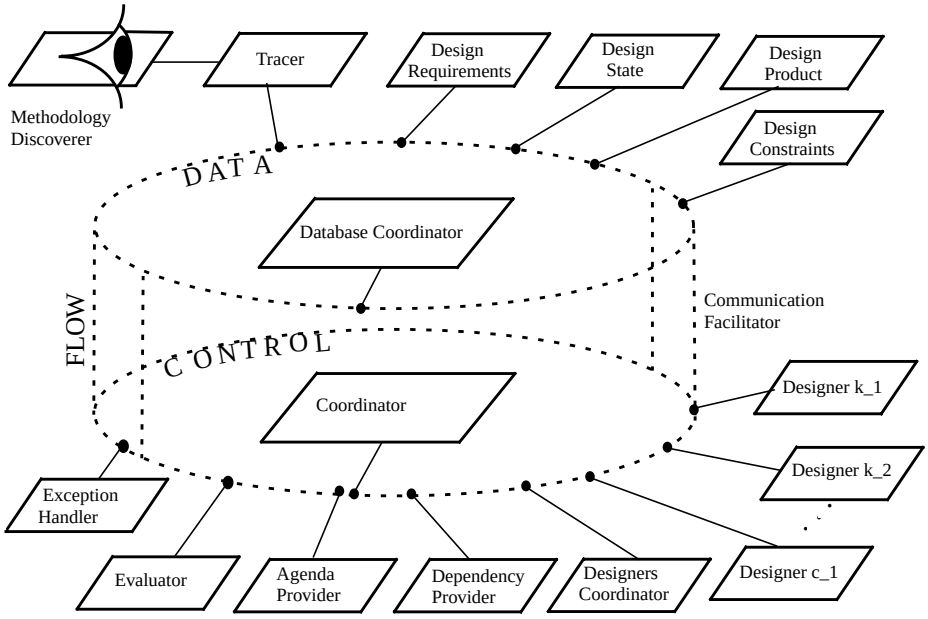


Figure 2. The Architecture of the Multi-agent Design System

The control of the flow of the design process that RD follows is based on the concept of *design cycles*. A design cycle starts when the set of designers at a specific depth in dependency graph are asked to design. At the end of each design cycle the results of the design are checked against the constraints. If the results satisfy all the relevant constraints the corresponding design cycle is interpreted as successful otherwise it is labeled as unsuccessful. Figure 4 shows how the design process is moved through a design cycle.

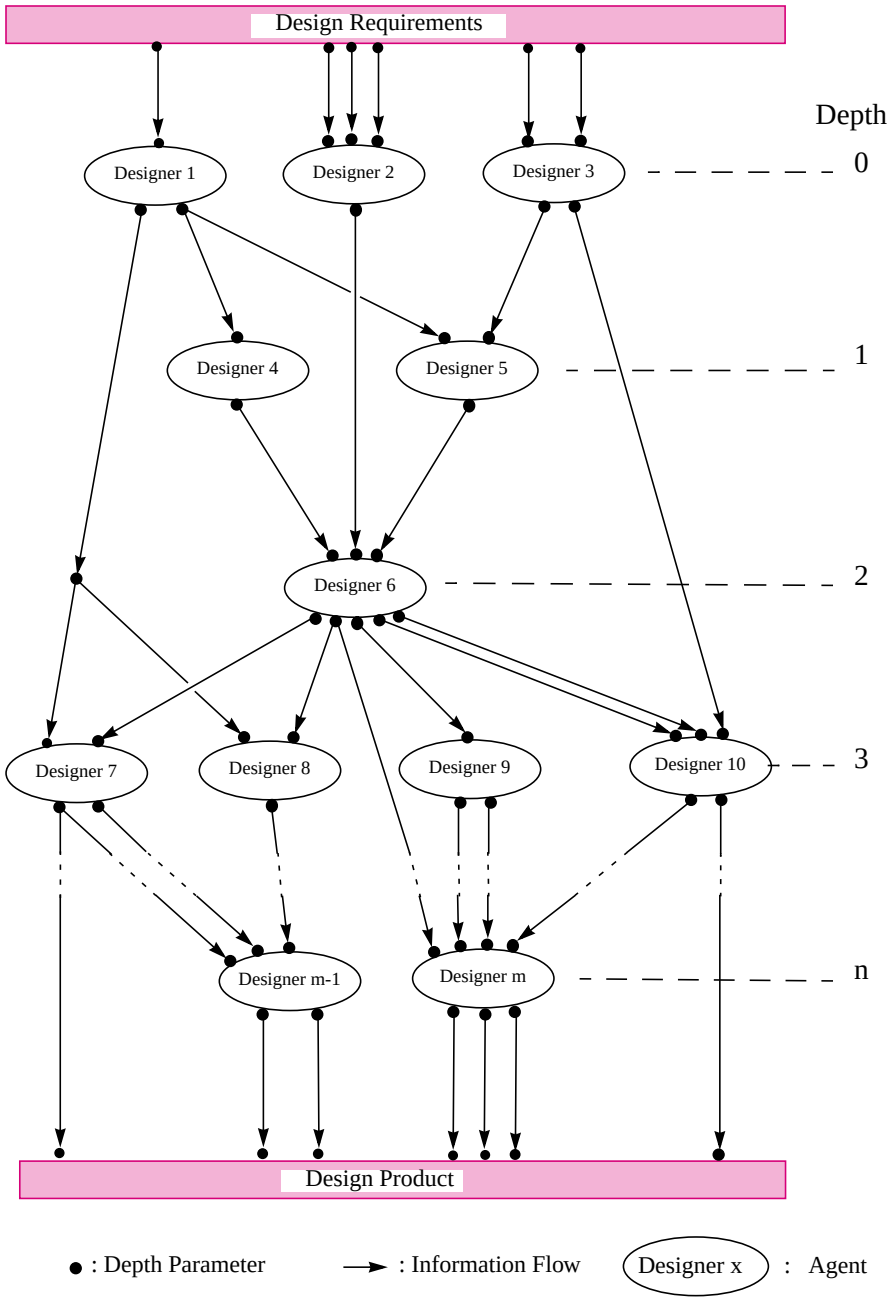


Figure 3. Dependency Graph

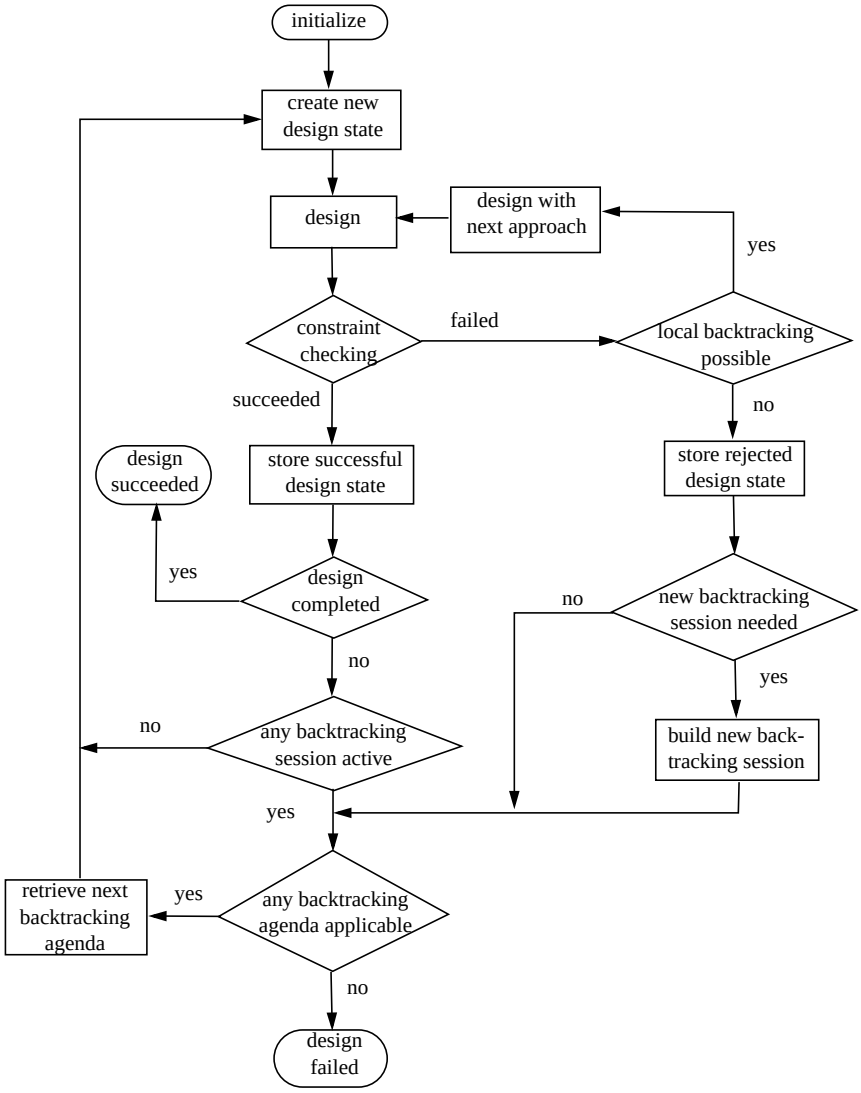


Figure 4. Flowchart of the Design Process.

5. The Experiments

Each designer agent in Figure 3 may have different approaches for generating its output design parameters. For generating a design, a combination of different design approaches from different designers are used. If the generated

design does not satisfy the constraints, another combination of design approaches is tried.

The knowledge about how designer agents are dependent on each other is used to select those paths that have a chance of resolving the constraint violation. They are executed while the rest will be pruned. This reduces the time and effort needed to find the path that generates a successful design (i.e., the design that satisfies all the constraints). This technique is known as dependency-directed backtracking.

Table 1 compares the two runs of the system for the same project one with dependency-directed and the other by exhaustive backtracking. It is clear that in this context dependency-directed backtracking is a superior method in terms of time and resources used to find a successful design.

TABLE 1. Advantage of Dependency-directed Backtracking

type of backtracking	trace index	number of cycles	time spent (hours)	memory size	number of events	number of messages
exhaustive	2118	5294	02:54:24	36921.0 K	2085879	341268
dependency	2118	2171	00:47:07	13304.0 K	867179	140069

The design process takes different paths through agents to generate different candidate solutions for the same set of requirements. The candidate solutions that satisfy the set of constraints are the acceptable designs. Figure 5 shows how selecting different design approaches produces different design paths.

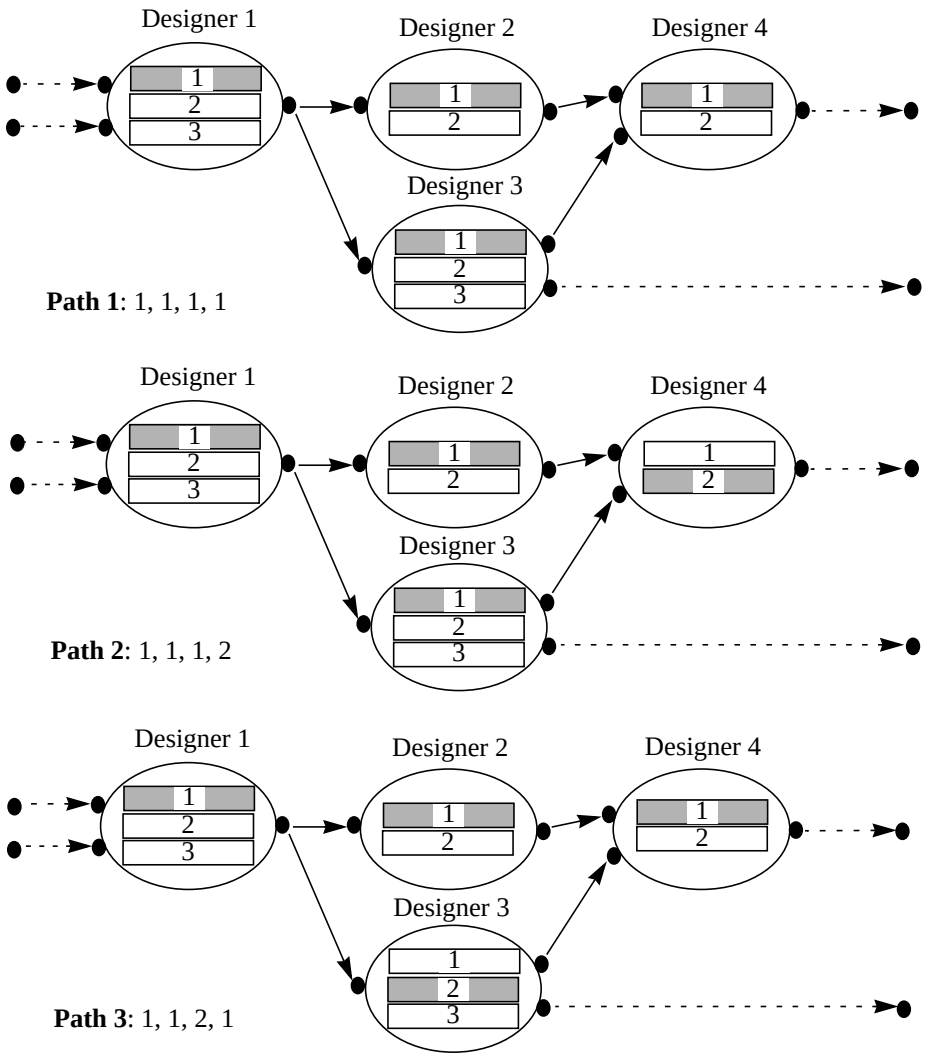


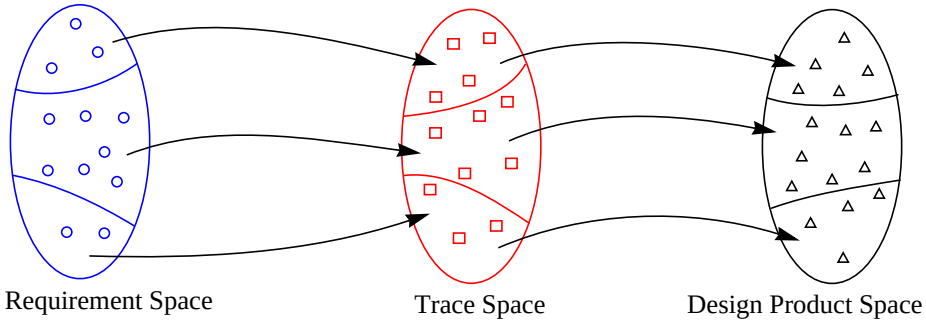
Figure 5. Different Design Paths

In Figure 5 a path is represented by the sequence of approach indices that were used, e.g., `1,1,1,2`. When a constraint is violated, designer agents systematically check all other possible design paths by varying their design approaches. By taking different paths the system leaves a *trace* behind which is recorded by the `Tracer` agent.

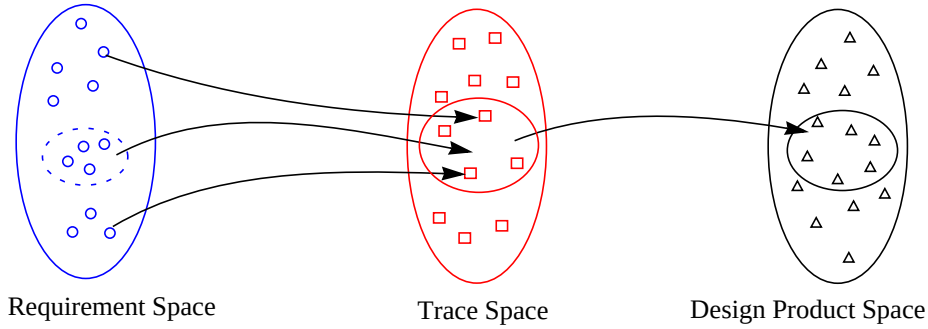
From an abstract point of view, the multi-agent design system, RD, maps the space of design requirements (i.e., problem space) to the space of traces and then to the space of designs (i.e., design products). The following scenarios may happen in mapping the requirement space to the design space (see Figure 6). A design “project” is a particular combination of requirements.

- **Case 1:** Each cluster of projects is mapped to the design space by exactly one cluster of traces.
- **Case 2:** A cluster of projects plus some exceptions not included in that cluster are mapped to the design space by exactly one cluster of traces.
- **Case 3:** A cluster of projects is mapped to the design space by one trace cluster plus some exception traces that do not fit in the cluster.

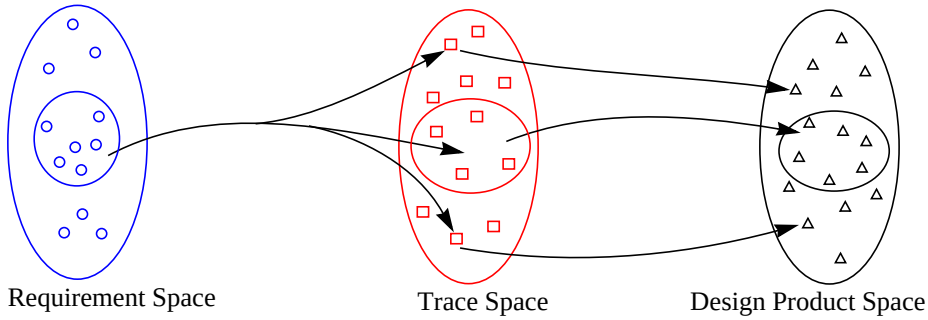
Other cases might occur that are a mixture of the above cases. However, with respect to generating methodologies, the most desirable cases are cases 1 to 3. The reason is that the above cases have the least exceptions, therefore the generalization becomes much cleaner and will cover more situations.



Case 1: Exact Match between Clusters



Case 2: Partial Match Includes Exceptions



Case 3: Partial Match Includes Exceptions

Figure 6. Some Different Scenarios in Mapping Requirements to Designs

Many sensitivity analyses were conducted in order to find the appropriate ranges of values for design requirements (Table 2). We limited the number of variations to keep the size of the design problems that need to be solved within a manageable range (e.g., 1000 problems). That is because the number of the problems is a function of the number of the variations of the requirements and constraints.

TABLE 2. Different Values for the Robot Arm Requirements.

`workspace`	{small-M, small-L, big-M, big-L}
`workload` (kg)	{1.0, 2.0, 3.0, 4.0, 5.0}
`settling_time` (sec)	{3.0, 2.0, 1.0}
`maximum_overshoot` (%)	{50, 40, 20, 10}

The idea of extracting design methodologies from traces is related to the subject of *concept formation* in Machine Learning. “Concept formation is the task of automatically inferring the general definition of some concept, given examples labeled as members or nonmembers of the concept” (Mitchell, 1997, p. 21).

In this work we used an approach for concept formation called Agglomerative Formation of Concept Hierarchies (ACH) (Langley, 1996, pp. 212-217) to find correlations between the requirement space and trace space. A clustering based on this correspondence allows the retrieval of an appropriate trace given a new set of requirements that is similar to an existing one.

6. The Results

We used RD to solve a set of 960 design projects. Figure 7 shows how many projects followed a specific trace. The promising results is that the distribution of the traces is quite scattered—that is, many projects followed similar traces. The total number of possible traces is the product of the number of design approaches of all the designer agents. For the experiments shown in Figure 7 the total number of possible traces is 2304. Among all 2304 possible traces only 84 were followed to generate successful designs, i.e., less than 4%.

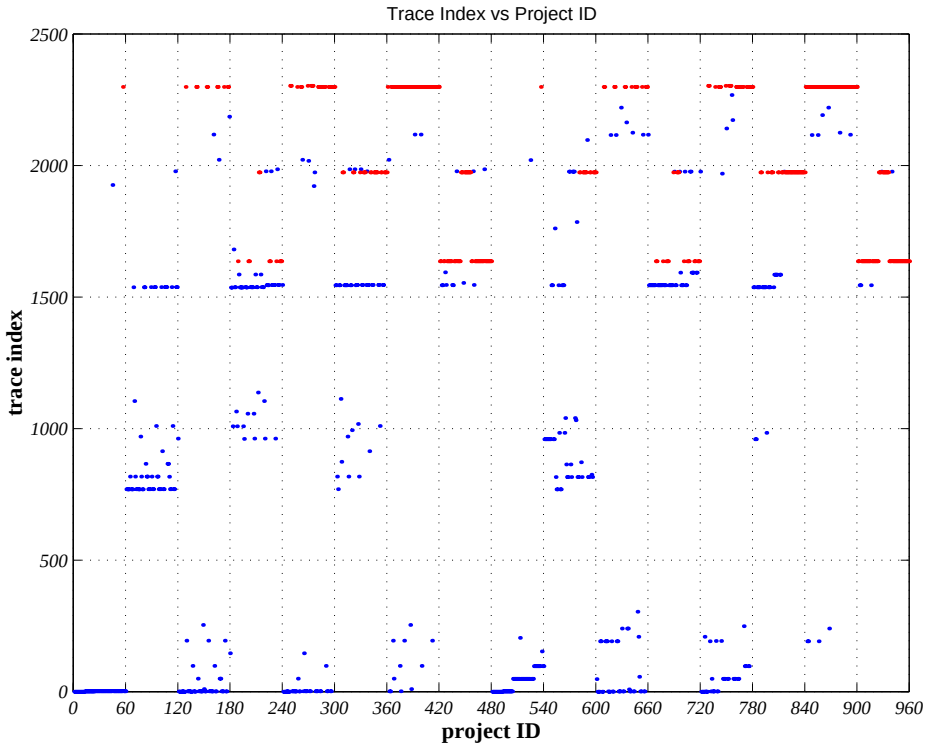


Figure 7. Distribution of Traces versus Projects

The low percentage of successful traces indicates that for each group of projects that followed a particular trace there is a unique combination of approaches leading to successful designs, hence there is a high chance that if similar projects follow the same trace they will succeed in generating successful designs. As a result, the path followed by those projects can lead us to formulating a design methodology for those projects as well as projects that are similar.

Figure 8 shows the frequency of all 84 successful traces. It is evident from this figure that a small number of traces have very high frequencies (say higher than 20). This is good news for being able to find design methodologies. A small number of traces with relatively high frequency shows that even without clustering traces together, we are able to find methodologies that are based on those traces and still cover a large number of different situations (e.g., 70 different projects).

Moreover, existence of a small number of traces with high frequencies

helps in clustering traces together. The high frequency traces can act as seeds for clustering—that is, they dominate the traces with lower frequencies and form generalized traces with even more frequencies.

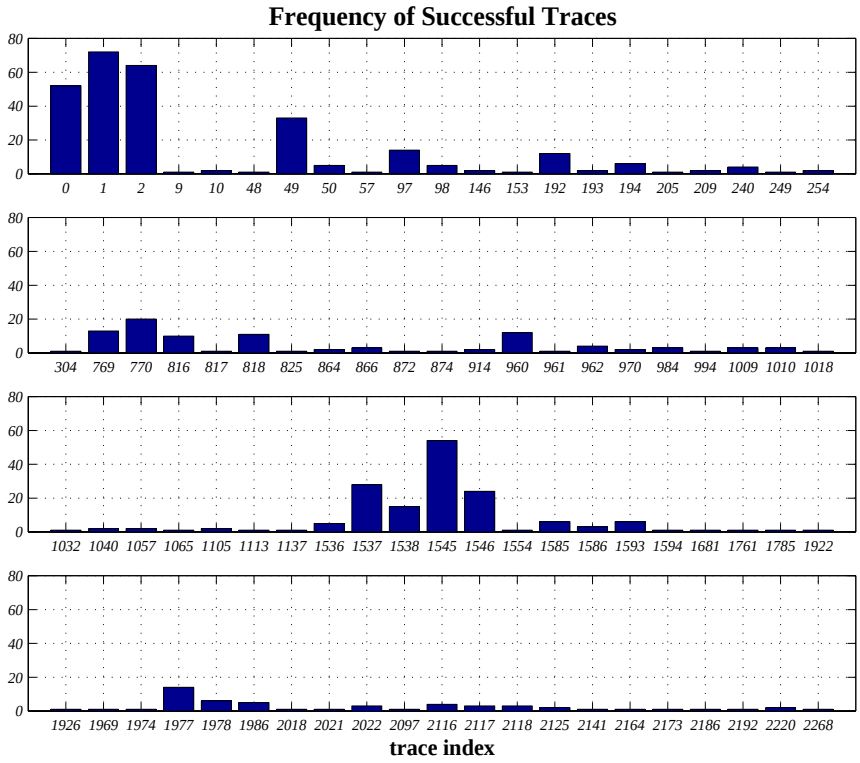


Figure 8. Frequency of Successful Traces.

The set of successful traces that are close enough can be clustered together to form a generalized trace. A generalized trace covers all the projects that followed each of the traces incorporated in the generalized trace. Design methodologies are formulated by extracting the correlation between a generalized trace and the design projects that followed that trace. The sample design methodology shown in Figure 1 is the English translation of the correlation between design projects and the corresponding traces.

7. Importance of this Research

Using system-developed methodologies allows effective and efficient practices to be used from the start of a project instead of being learned from experience. These new methodologies are radically different from the sequential,

discipline-based ones. They also reduce time-to-market and save resources. To be able to compete in today's global market, companies need continuous improvements in product quality and improvements to the performance of their design and manufacturing processes. Integration reduces the number of failures and backtracking by facilitating information sharing, thus saving resources and reducing design time. Integration also provides collaboration between different participants that, as a result, enhances the quality of the design.

Agent-based systems allow the incorporation of new technologies systematically and quickly through the addition or deletion of agents. Thus new knowledge can be added, and old knowledge removed rapidly. Running the system with the new set of agents will result in new traces and thus new and different methodologies. In addition, design processes can be biased toward more environmentally friendly products, as the alternative methods that are built into each agent are tried in a preferential order, and as each method tends to contribute differently towards the final properties of the design.

The research attacks the problem of integration in multi-disciplinary design. The number of specialists is increasing, while the number of generalists, capable of doing system integration, is decreasing. Also the knowledge burden on the designer keeps increasing due to more materials and more options (NSF, 1996). Thus it is becoming harder to develop methodologies for the integration of multiple disciplines in design. An increasingly specialized technological environment tends to force designers to concentrate on some disciplines more than others. This research allows designers to see the whole design problem.

Computers have mostly been used to support the manipulation and analysis of design product information. This work focuses on the design process, an aspect that has not benefited from computers very much. It applies computers to new areas of engineering design by incorporating new software methods.

Simulation of design processes based on a multi-agent paradigm is a new area of research that has a high potential for practical as well as theoretical impact on the design of products. The use of multi-agent systems technology is growing rapidly with the development of Java-based systems and agent access across the world-wide web.

We have also taken advantage of many AI techniques, and have based our work on previous research in the area of AI in Design. The research incorporates judgement and experience. "System integration, many consider, is an ill-structured problem... No specific rules have to be followed when doing integration... Experienced designers deal with system integration using judgement and experience. Knowledge-based programming technology offers a method-

ology to tackle these ill-structured integration and design problems" (Sobolewski, 1996).

This work benefits from inter-disciplinary contributions from both Artificial Intelligence and Engineering Design. According to NSF's report (1996) on Research Opportunities in Engineering Design, "research areas that will have greatest impact on engineering design over the next 10 years are: Collaborative Design Tools and Techniques, Perspective Models/Methods, System Integration Infrastructure/Tools, and Design Information Support Systems". This work covers all of these areas of research and hence is expected to have a strong impact.

8. Conclusions

The potential applications of this research are in multi-disciplinary design situations, such as those that occur throughout the automotive or aerospace industries, where large gains can be achieved with integrated methodologies. New methodologies can be customized so that they are biased toward specific objectives such as manufacturability or being environmentally friendly. In addition, current methodologies can be analyzed for flaws and bottlenecks, and necessary refinements made.

By applying this approach the response time for the incorporation of new technologies in design processes can be reduced. Methodologies can be refined as soon as a change occurs in the market or in the organization of the company.

This research has shown that the following hypothesis is true: *Computers can provide us with better ways of doing design by discovering superior design methodologies that integrate different points-of-view of multiple disciplines in the design process.*

It is possible to use computers to simulate the design process. We can then analyze the results of the simulation to synthesize design methodologies that have superior features. This research has produced convincing, preliminary results. The approach that we have proposed has been developed based on parametric design problems. Applicability of the approach to other types of problems needs to be investigated.

9. References

BROWN, D. C., S. E. LANDER & C. J. PETRIE, "The Application of Multi-agent Systems to Concurrent Engineering", Editorial: *Concurrent Engineering: Research and Applications*, Technomic Publ., V.4, N.1, (March 1996), pp. 2-5.

DUFFY, A. H. B., "The 'What' and 'How' of Learning in Design", *IEEE Expert: Intelligent Systems & Their Applications*, Vol. 12, No. 3, (May-June 1997), pp. 71-76

GEBALA, D. G. and S. D. EPPINGER, "Methods for Analyzing Design Procedures", *Proceeding of ASME Design Theory and Methodology Conference: DTM'91*, ASME DE-Vol. 31, (1991), pp. 227-233.

HALE, M., J. I. CRAIG, F. MISTREE, D. P. SCHRAGE, "DREAMS and IMAGE: A Model and Computer Implementation for Concurrent, Life-Cycle Design of Complex Systems", *Concurrent Engineering: Research and Applications*, Vol. 4, No. 2, (June 1996), pp. 171-186.

KROO, I. and M. TAKAI, "Aircraft Design Optimization Using a Quasi-Procedural Method and Expert System", *Multidisciplinary Design and Optimization Symposium*, (Nov. 1990).

LANDER, S., "Issues in Multi-agent Design Systems", *IEEE Expert: Intelligent Systems and their Applications*, (March-April 1997), Vol. 12, No. 2, pp. 18-26.

LANGLEY, P., *Elements of Machine Learning*, Morgan Kaufmann, Inc., (1996).

LEVITT, R. E., Y. JIN, and C. L. DYM, "Knowledge-Based Support for Management of Concurrent, Multidisciplinary Design", *AI EDAM*, Academic Press Ltd., Vol. 5, No. 2, (1991), pp. 77-95.

LIU, J., and D. C. BROWN, "Generating Design Decomposition Knowledge for Parametric Design Problems", *AID-94: Artificial Intelligence in Design'94*, (Eds.) Gero & Sudweeks, Kluwer Academic Publ., (1994), pp. 661-678.

MITCHELL, T. M., *Machine Learning*, McGraw-Hill Companies, Inc., (1997).

NATIONAL SCIENCE FOUNDATION, *Research Opportunities in Engineering Design*, NSF Workshop Final Report, (April 1996).

RIVIN, E. I., *Mechanical Design of Robots*, McGraw-Hill Book Company, (1988).

SHAKERI, C., *Discovery of Design Methodologies for the Integration of Multi-disciplinary Design Problems*, Ph.D. Thesis, Mechanical Engineering Department, WPI, Worcester, MA 01609, USA, (December 1998).

SHAKERI, C., D. C. BROWN, and M. N. NOORI, "Discovering Methodologies for Integrated Product Design", *Proceedings of 1998 AI & Manufacturing Workshop: State of the Art and State of Practice*, (Aug. 31-Sept. 2, 1998), Albuquerque, NM, pp. 169-176.

SOBOLEWSKI, M., "Multiagent Knowledge-Based Environment for Concurrent Engineering Applications", *Concurrent Engineering: Research and Applications*, Technomic Publ., V.4, N.1, (March 1996), pp. 89-97.

SOBIESZCZANSKI-SOBIESKI, J. and R. T. HAFTKA, "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments", *34th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, AIAA Paper No. 96-0711, (January 1996), p. 32.

WOOLDRIDGE, M., "Agent-based software engineering", *IEEE Transactions on Software Engineering*, (1997), 144(1): 26-37.

WOYAK, S., B. MALONE, and A. MYKLEBUST, "An Architecture for Creating Engineering Applications: The Dynamic Integration System", *Proceeding of ASME Computers in Engineering Conf.*, Boston, MA, (September 1995).

WUJEK, B. A., J. E. RENAUD, S. M. BATILL, and J. B. BROCKMAN, "Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment", *Concurrent Engineering: Research and Applications*, Vol. 4, No. 4, (December 1996), pp. 361-377.