A Constraint-Aided Conceptual Design Environment for Autodesk Inventor *

Alan Holland, Barry O'Callaghan, and Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{a.holland|b.ocallaghan|b.osullivan}@4c.ucc.ie

Abstract. Engineering conceptual design can be defined as that phase of the product development process during which the designer takes a specification for a product to be designed and generates many broad solutions for it. It is well recognized that few computational tools exist that are capable of supporting the designer work through the conceptual phase of design. However, significant recent developments have been made in solid modeling and 3D computer-aided design. The use of such tools has become a critical element in the more sophisticated product development processes to be found in modern industry. This paper presents a prototype constraint-based computer-aided design (CAD) technology that can be used to support designers working in the early stages of design. The technology has been developed as an add-in application for Autodesk Inventor, a 3D solid-modeling environment. The add-in has, at its core, a constraint filtering system based on generalised arc-consistency processing and backtrack search. We present our current prototype and a detailed demonstration of its functionality. Finally, we describe our current work on a number of additional features for the next prototype, which will be deployed in an industrial context.

1 Introduction

Engineering conceptual design can be regarded as that phase of the engineering design process during which the designer takes a specification for a product to be designed and generates many broad solutions for it. Each of these broad solutions is generally referred to as a *scheme* [7]. It is generally accepted that conceptual design is one of the most critical phases of the product development process. It has been reported that more than 75% of a product's total cost is dictated by decisions made during the conceptual phase of design and that poor conceptual design can never be compensated for at the detailed design stage [10].

To support interactive conceptual design a number of issues must be considered. Firstly, the conceptual design process is initiated by a statement describing the desired properties of the required product. This statement may not be complete and may be modified during design. Secondly, conceptual design is a process in which synthesis of a scheme is a fundamental activity. However, the designer should have the freedom to

* This research is funded by Enterprise Ireland, through their Research Innovation Fund (Grant Number RIF-2001-317). The software used for the project, Autodesk Inventor, has been sponsored by cadcoevolution.com, an Irish CAD provider.

approach the process in anyway he wishes. Thirdly, in so far as it is possible, designers should be alerted to any inconsistencies that exist in their designs. Designers may seek explanations for such inconsistencies, or justifications for why certain options are not available to them. Alternatively, designers may wish to be given an explanation detailing how a particular scheme has come about. Fourthly, automated evaluation and comparison of multiple schemes, throughout the design process, is necessary to focus the designer on promising alternatives. Finally, designers prefer to use tools which are familiar to them and, therefore, any additional tools that a designer is expected to use must have a "look-and-feel" similar to those they already use. It was these considerations that set the agenda for the work reported here.

This paper presents a prototype constraint-based computer-aided design (CAD) technology that can be used to support designers working in the early stages of design. We present the current status of our prototype and a detailed demonstration of its functionality. We report on some industrial experiences and describe our current work on a number of additional features for the next prototype, which will be deployed in an industrial context. The CAD technology has, at its core, a constraint filtering system based on generalised arc-consistency processing [3] and backtrack search. Using the technology, the designer can be assisted in developing and evaluating a set of schemes which satisfy the various constraints that are imposed on the design. Explanations and justifications can be generated to aid the designer's understanding of the state of the design problem. Arbitrary constraints can be asserted or retracted by the designer which permits the incorporation of new requirements into the design specification and give the designer freedom to approach the process as he wishes.

The remainder of the paper is organized as follows. Section 2 presents an overview of the relevant literature. Section 3 presents an overview of the theory of conceptual design upon which the work presented in this paper is based. Section 4 presents an overview of the current prototype of our CAD technology for supporting conceptual design. Section 5 outlines our plans for future development and deployment. In Section 6 a number of concluding remarks are made.

2 Related Research

In the design literature three phases of design are generally identified: conceptual design, embodiment design and detailed design [16]. During conceptual design the designer searches for a set of broad solutions to a design problem, each of which satisfies the fundamental requirements of the desired product. The embodiment phase of design is traditionally regarded as the phase during which an initial physical design is developed. This initial physical design involves the determination of component arrangements, initial forms and other part characteristics. The detailed phase of design is traditionally regarded as the phase during which the final physical design is developed.

Constraint-based applications for design have been more commonly applied to the post-conceptual phases of design [11, 12, 21]. The use of constraint processing techniques for supporting configuration design has also been widely reported in the literature [13, 19].

Modern approaches to product development, such as Concurrent Engineering [4], attempt to maximize the degree to which design activities are performed in parallel. Researchers in the constraint processing community have developed constraint-based technologies that support integrated approaches to product development [5]. Constraint-based approaches to managing conflict in collaborative design systems have also been reported [2, 9, 11]. Using constraints to co-ordinate distributed agents in engineering design has also been reported [17].

Constraint-based approaches to supporting conceptual design have been reported in the literature for quite a number of years [8, 18, 20]. However, most of this research does not address the synthesis problem; the vast majority has focused on constraint propagation and consistency management relating to more numerical design decisions. For example, "Concept Modeler" is based on a set of graph processing algorithms that use bipartite matching and strong component identification for solving systems of equations [20]. The Concept Modeler system allows the designer to construct models of a product using iconic abstractions of machine elements.

Based on the earlier work on Concept Modeler, a system called "Design Sheet" has been developed [18]. This system is essentially an environment for facilitating flexible trade-off studies during conceptual design. It integrates constraint management techniques, symbolic mathematics and robust equation solving capabilities with a flexible environment for developing models and specifying tradeoff studies. The Design Sheet system permits a designer to build a model of a design by entering a set of algebraic constraints. The designer can then use Design Sheet to change the set of independent variables in the algebraic model and perform trade-off studies, optimization and sensitivity analysis.

While not a constraint-based system, the Conceptual Understanding and Prototyping Environment (CUP) is an approach to supporting conceptual design that unites ideas from traditional mechanical design with 3D layouts and knowledge engineering [1]. Of all of the systems reviewed here, CUP is most similar to the approach that we have adopted. However, our technology is entirely constraint-based which gives us the opportunity to exploit the semantics of constraints and use inference as a core technique for navigating the design search space, providing explanations and an immediate declarative approach to modeling the evolving schemes that the designer wishes to explore.

3 The Approach

In this section we will give an overview of the approach to conceptual design that has motivated the design of the CAD system which we will present in Section 4. The model of conceptual design adopted here is based on the well accepted observation that during this phase of design a designer works from an informal set of requirements that the product must satisfy and generates alternative schemes which satisfy them.

Central to the process of scheme generation is an understanding of function and how it can be provided. The process involves the development of a functional decomposition which provides the basis for a realization of physical elements that form a scheme. In addition to determining which physical elements comprise a scheme, the relations between them must also be specified to a sufficient extent to permit the evaluation and comparison of alternative schemes.

In the remainder of this section a brief overview of some of the most important aspects of our approach to conceptual design will be presented. For a more complete discussion of the theory the reader is encouraged to refer to the more detailed literature available [14, 15].

3.1 The Design Specification

The conceptual design process is initiated by the recognition of a need or customer requirement. This need is analyzed and translated into a statement which defines the functionality that the product should provide (referred to as *functional* requirements) and the *physical* requirements that the product must satisfy. This statement is known as a *design specification*.

A design specification will contain both abstract functional requirements as well as concrete physical requirements. The functional requirements define the "purpose" of the desired product at as high a level of abstraction as possible. In addition, two classes of physical requirement can be identified: product requirements and life-cycle requirements. These requirements can be either *categorical requirements* that define constraints between attributes of the product or its life-cycle, or they can be *preferences* related to subsets of these attributes.

Essentially, the design specification comprises a set of constraints which must be satisfied and a set of objective functions with respect to which the desired product must be Pareto optimal [14].

3.2 Conceptual Design Knowledge

In order to successfully synthesize alternative schemes that meet the requirements defined in the design specification, the designer needs considerable knowledge of how function can be provided by physical means. This knowledge exists in a variety of forms; a designer may not only know of particular components, sub-assemblies and technologies that can provide particular functionality, but may be aware of abstract concepts which could also be used. For example, a designer may know that an electric light-bulb can generate heat or, alternatively, that heat can be generated by rubbing two surfaces together. The latter concept is more abstract that the former. In order to effectively support the designer during conceptual design, these alternative types of design knowledge need to be modeled in a formal way. However, a CAD system which supports conceptual design must also be capable of dealing with the designer's "off-the-cuff" ideas and store them for future use if they are deemed useful.

Reasoning about Function We employ a *function-means map* approach to cataloging how function can be provided by means [6, 14]. In a function-means map two different types of means can be identified: *design principles* and *design entities*. A design principle is a means which is defined in terms of a set of functions that must be provided in order to provide some higher-level functionality. Design principles are abstractions of known approaches to providing function. By utilizing a design principle the designer can decompose higher-level functions without committing to a physical solution too early in the design process. The functions that are required by a design principle collectively replace the function being embodied by that principle. The functions which define a design principle will, generally, have a number of *context relations* defined between them. These context relations describe how the parts in the scheme, which provide these functions, should be configured so that the design principle is used in a valid way. For example, in Figure 1, *principle 1* comprises two functions, *f1* and *f2*, between which a context relation *r1* is defined.

Note that a design principle is not just a model of a known physical design solution, but is an abstraction which can be used to encourage creativity and analogical reasoning during design. An example of this was presented above when defining an abstract concept for generating heat by rubbing two surfaces together.

A design entity, on the other hand, is a physical, tangible means for providing function such as a component or sub-assembly. A design entity is defined by a set of parameters and the constraints that exist between these parameters. For example, an electronic resistor would be modeled as a design entity which is defined by three parameters, resistance, voltage and current, between which Ohm's Law would hold.

Embodiment of Function As the designer develops a scheme for a product every function in the scheme is embodied by a means. Each means that is available to the designer has an associated *set of behaviors*. Each *behavior* is defined as a set of functions that the means can be used to provide simultaneously. Each behavior associated with a design principle will contain only one function to reflect the fact that it is used to decompose a single function. However, a behavior associated with a design entity may contain many functions to reflect the fact the there are many combinations of functions that the entity can provide at the same time. For example, the bulb design entity mentioned earlier may be able to fulfill the functions *provide light* and *generate heat* simultaneously. However, when a design entity is incorporated into a scheme it is not necessary that every function in this behavior be used in the scheme.

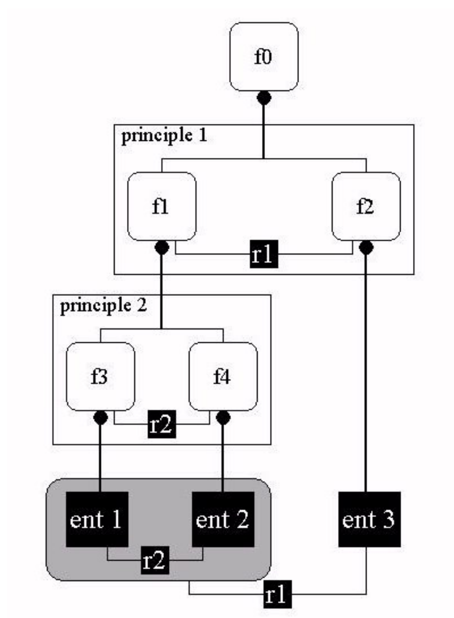

Fig. 1: An example of scheme configuration.

3.3 Scheme Configuration using Interfaces

Generally, the first means that a designer will select will be a design principle. This design principle will substitute the required (parent) functionality with a set of child functions. Ultimately the designer will embody all leaf-node functions in the functional decomposition with design entities. During this embodiment process, the context relations from the design principles used in the scheme will be used as a basis for defining the interfaces (constraints) between the design entities used in the scheme.

For example, in Figure 1 an example scheme structure is illustrated. The top-level function in this scheme is *f0*. This function is embodied using a design principle called *principle 1*. This design principle introduces two functions, *f1* and *f2* to replace the function *f0*. A context relation, *r1*, is specified between these functions. The function *f1* is embodied by a design principle, *principle 2*, which introduces two further functions, *f3* and *f4* into the scheme. A context relation, *r2*, is specified between these functions. The function *f3* is embodied with the design entity *ent 1*, the function *f4* is embodied with the design entity *ent 2* and the function *f2* is embodied with the design entity *ent 3*. However, between which design entities should the context relations *r1* and *r2* be considered?

The context relation *r2* must exist between the entities that derive from the functions *f3* and *f4*. The design entities *ent 1* and *ent 2* are used to embody these functions. Thus, the context relation *r2* must be considered between these entities. Since, the design entities *ent 1* and *ent 2* are the means used to provide the functions *f3* and *f4*, these entities can be regarded as being *directly* used to provide these functions.

The context relation *r1* is a little more complex. This context relation must exist between the entities that derive from the functions *f1* and *f2*. The design entity *ent 3* is used to embody the function *f2*. Thus this entity can be regarded as being *directly* used for the function *f2*. The function *f1* is provided by the design principle *principle 2*, whose child functions are in turn embodied by the design entities *ent 1* and *ent 2*. Thus, these design entities can be regarded as being *indirectly* used to provide the function *f1*. Therefore, the context relation *r1* must be considered between the combination of design entities *ent 1* and *ent 2* on the one hand, and *ent 3* on the other hand.

The precise nature of these interfaces cannot be known with certainty until the designer embodies functions with design entities; this is because the link between functions and design entities is generally not known with certainty during the development of the functional decomposition for the scheme. Until the precise nature of a particular interface is known, they are modeled as constraints between design entities which can be used to reason about the product structure; for example, interfaces may represent simple spatial relationships which can inform an evaluation related to the relative position of parts in a product.

The types of interfaces that may be used to synthesize a product structure will be specific to the engineering domain within which the designer is working. Indeed, these interfaces may also be specific to the particular company to which the designer belongs in order to ensure the configurability of the product.

4 An Overview of the Current Prototype

In this section we present the key features of our current prototype CAD technology for supporting conceptual design, which we call ConCAD Expert. The technology is seamlessly integrated with Autodesk Inventor[1]. This particular CAD system has been chosen for a number of reasons. In particular, as well as being one of the most popular 3D solid modeling design environments, Inventor has an architecture similar to most tools of its kind, but has a very rich API through which we can integrate with the host CAD system.

Our technology has been designed to interface to the Inventor (version 5.3) CAD system as an add-in application that can be invoked by the designer at any point. At its core is an interactive constraint filtering system based on generalised arc-consistency [3] and backtrack search. The system is fully interactive, monitoring the consistency of designer decisions and providing feedback when an inconsistency has been detected or the designer has requested justifications or explanations from the system. It was developed using C#, and uses the Autodesk Inventor 5.3 COM API. An XML database has been developed to store the parts available to the CAD system. The XML schema that has been used, represents a part file and the various attributes of each part. We now present many of the features of our prototype in the form of a "walk-through".

Once the conceptual design tool has been invoked, the designer can use a Functional View panel (see left-hand side of Figure 2) to define the functionality and physical constraints that the desired product must satisfy.

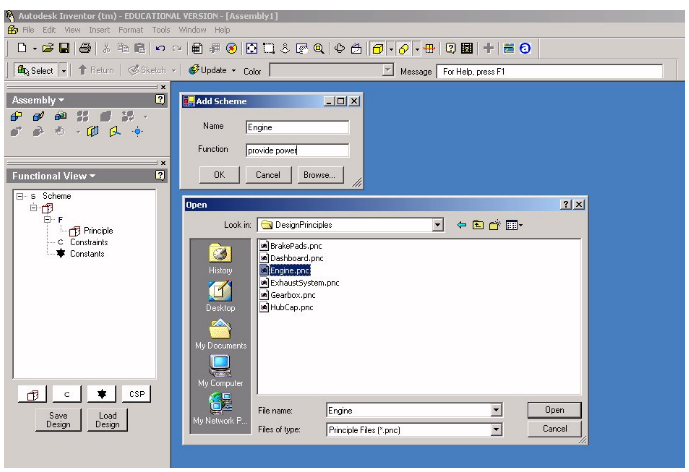

Fig. 2: Beginning a conceptual design session using a design principle.

Once the ConCAD Expert add-in has been invoked the designer can begin to develop his concept by developing a functional decomposition and mapping functions

[1] See: `http://www.autodesk.com/inventor`

onto parts which will be loaded into Inventor automatically. In Figure 2 the designer begins to design an engine by incorporating a pre-defined design principle for an internal combustion engine. The result of this action can be seen in Figure 3, where each of the functional elements of an engine can be seen in the Functional View pane. Note that this is an abstract description. Each of the elements in the Functional View pane relates to a design element which is represented as a function. No parts have yet been selected at this point. However, the designer is free to select parts if desired, or can extend the functional description of each element further by incorporating additional design principles into the scheme. Furthermore, the designer can define context relations (constraints) between elements of the functional description, in addition to those that form part of the definition of the engine design principle, if desired. These will define how each of the elements must relate each other in the part model.

At the moment, the designer can select design principles from a predefined database, or can manually define them on-the-fly. Obviously, we would like the conceptual design system to store any new descriptions for use in future projects. At the moment, this must be done by manually including them in the database. However, in the next version of the system the identification and storage of design principles will be done automatically. In this way the CAD system will acquire the design principles for a particular design domain over time.

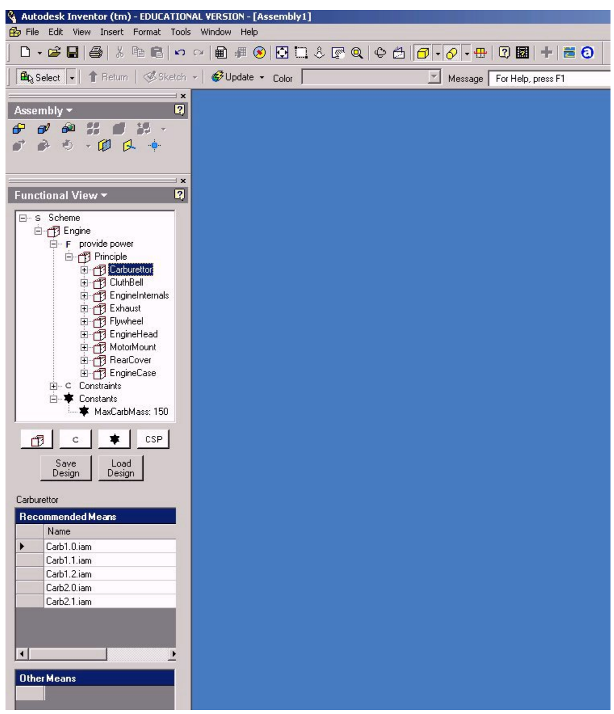

Fig. 3: Viewing recommended means.

At any point in time the designer can view which means in the CAD database are recommended for use in the designer's scheme. Recommended means must not only satisfy the functional requirements defined in the functional decomposition, but must also be consistent with the physical constraints defined explicitly by the designer and implicitly by any other means used in the scheme.

In Figure 3 the designer has asked the CAD system to recommend means for the carburettor element in the functional decomposition. As can be seen on the middle-left side of the figure, five alternative means have been found for this element from which the designer is free to choose. In this example, all means that can provide the required functionality for this element are recommended; otherwise they would have been displayed in the *Other Means* box. A means is no longer recommended if it satisfies the functional constraints, but violates one or more physical constraints.

However, for the purposes of explanation lets assume that before the designer selects one of these means he defines an additional constraint in the CAD system. An interface used to do this is shown in Figure 4. The designer is free to define constraints on any characteristic of the scheme. In this figure, the designer has defined the constraint that ensures that the mass of the means that will be used to provide the functionality of the carburettor will have a mass that is no more than a given maximum value.

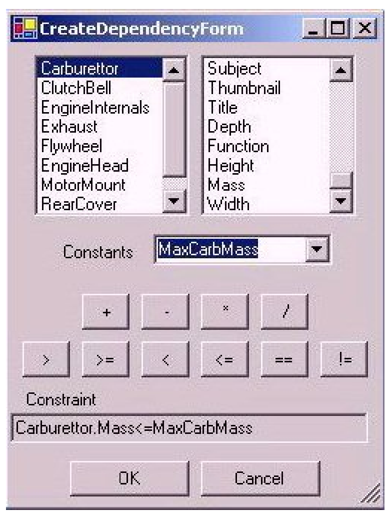

Fig. 4: Specifying additional constraints.

The consequence of this constraint is that some of the means which could have been used as a carburettor have been removed and placed in a list of *Other Means*, which contains those means that are no longer recommended for this situation. This scenario is depicted in the lower left of Figure 5.

Also presented in this figure is a window showing the attributes of a particular means selected by the designer. In this figure the designer has clicked on the recommended part, *Carb2.1.iam*. Note in the attributes window that the mass of the part satisfies the constraint that the maximum mass of the carburettor be no more than 150g.

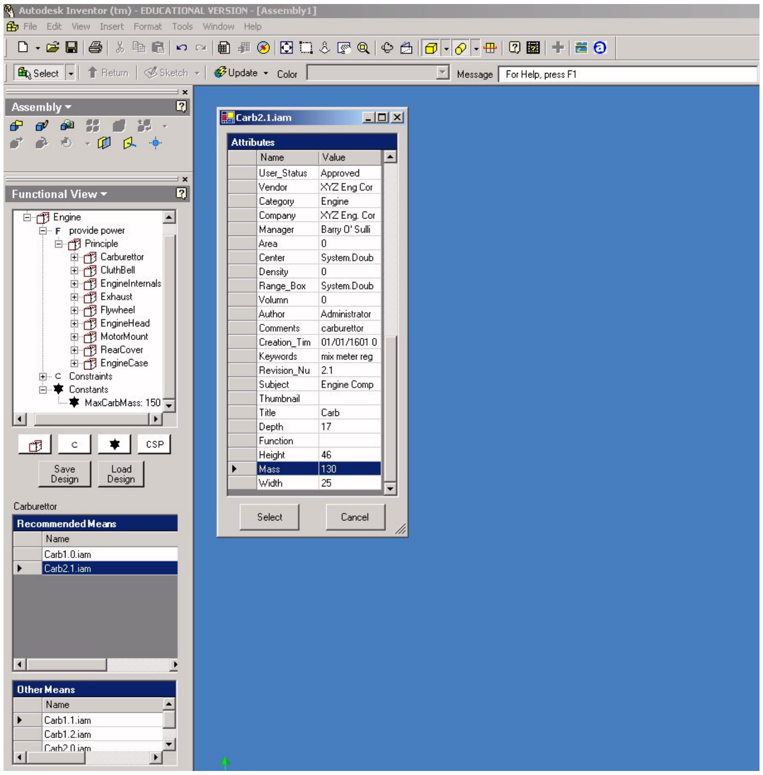

Fig. 5: The effect of the designer's constraint.

As the designer makes decisions and interacts with the CAD system, a CSP model describing the characteristics of the desired product is being developed. Part of the constraint model that the designer has developed at this point in our example is presented in Figure 6.

In this figure we can, firstly, see at the top the set of constant definitions. Secondly, we can see the set of variables in our design and their corresponding domains. Finally, we have the set of constraints defined by the design specification for the product, any additional constraints defined by the designer or constraints added to the model which are implied by the designer's decisions.

To illustrate the consequences of constraint addition and domain filtering in our prototype, Figure 7 shows the effect of adding an overall cost constraint to the model. This constraint was entered using the interface presented in Figure 4. We can see from Figure 7 that the list of of recommended means has been reduced further through in-

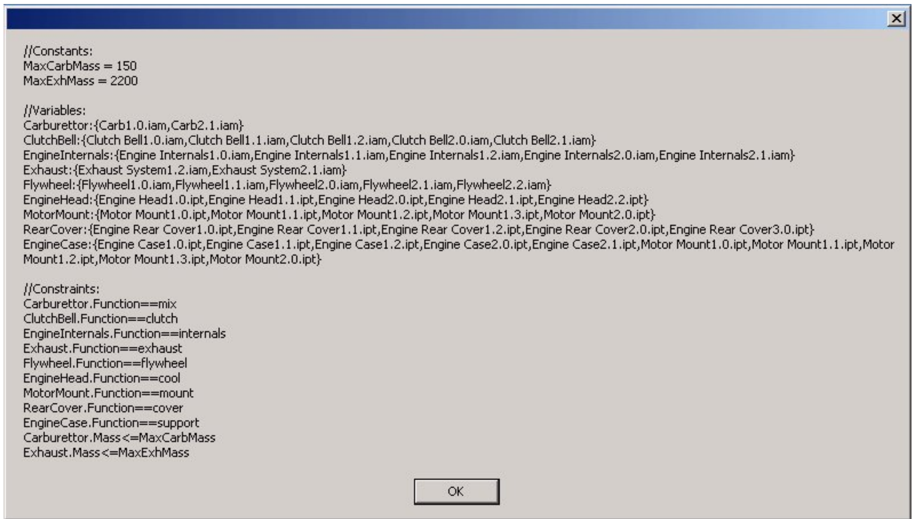

Fig. 6: The CSP can be viewed at any stage during the design process.

ference, which in the prototype is based on generalized consistency processing. In this figure the designer has asked for an explanation for why the *Carb1.0.iam* is no longer recommended by selecting it from the list. An explanation is given in Figure 8 which specifies the constraints with which the part is inconsistent. Explanations are currently generated directly from the inference algorithm used in the prototype.

Figure 9 presents the extended CSP model of our scheme which includes the designer-specified cost constraint (shown last in the figure). Note how the domains of the more expensive parts were heavily reduced whereas the smaller cheaper parts such as the *mount* were unaffected.

As the designer develops the functional decomposition he can also select parts for providing the necessary functionality. There is no restriction placed on the order that the designer develops a scheme. Figure 10 shows the CAD system interface after the designer has begun to select parts. The constraint model that the CAD system contains at this point not only includes constraints on the functional decomposition and each part, but also constraints on the way parts can be configured. These interface constraints derived from the constraints that the designer specified while developing the functional decomposition for the product, as well as the various design principles that were employed. As the designer develops the scheme further, inference on these constraints will assist the designer make consistent decisions on a valid configuration of these parts.

The designer does not need to fully specify the entire design at this point since that is a detailed design task. All that is required is that the designer specify the scheme to an extent which permits it to be evaluated against the constraints in the design specification and compared against any alternative schemes that have been developed.

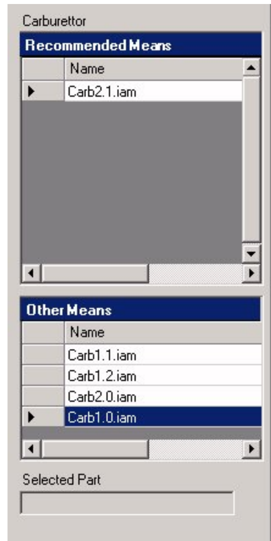

Fig. 7: A constraint was added setting the total Cost below a certain threshold. The Carb1.0.iam assembly violated this constraint because it was too expensive.

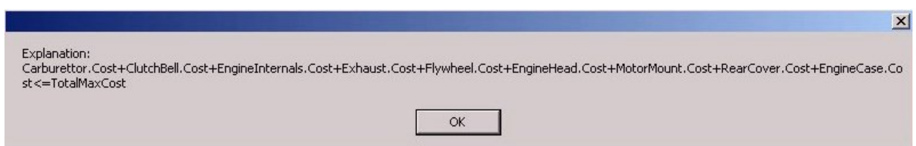

Fig. 8: An explanation showing the cost constraint that the the Carb1.0.iam assembly has violated.

5 Next Steps

The CAD technology presented here is being developed for industrial deployment. In anticipation of the release of the tool, we are currently developing the next generation of the CAD system which will have a number of additional capabilities. Some of these are discussed below.

In the current prototype the designer either manually enters the entire functional specification for an artifact, or uses predefined principles. We are currently working on an approach which helps capture new design principles and modifications of existing ones, and store them in the database for future use. In this way, not only will the CAD system be capable of learning from the designer, but will also alleviate a company from having to completely specify a design knowledge-base before the CAD technology is useful. The approach we are adopting is designer-driven. The designer decides what constitutes a sufficiently detailed abstract description of a principle. The system facilitates the designer by storing his concepts, whatever they may be. This feature is critical

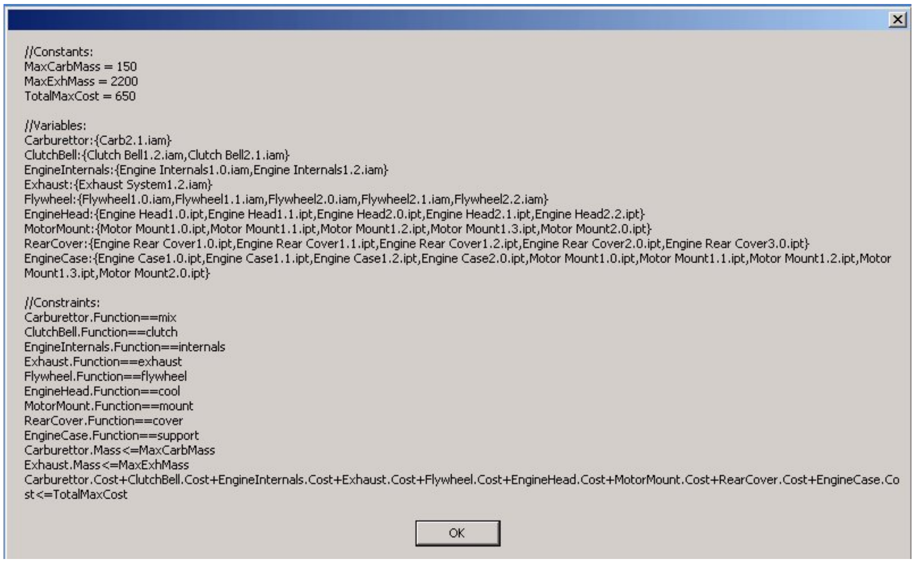

Fig. 9: The CSP after the overall cost constraint was added.

for real-world deployment; the need to invest heavily in knowledge-base development is a significant disadvantage of many intelligent CAD tools.

In addition to the above, we wish to extend the functional reasoning capabilities of the CAD system in a number of ways. In particular, we are developing a set of standard interface constraints to assist with reasoning about scheme configuration. For example, the ability to add spatial constraints is important in order to more accurately evaluate schemes. There are a number of other interface constraints that also need to be implemented as primitives, such as mechanical and electrical connections which have some default semantics. At present the designer does not have access to such default interfaces and must define them explicitly within the solid modeling environment of the host CAD system.

At present the functional description of a product is defined textually. We are interested in developing a sketch-based interface for describing the critical functional elements and their relationships. This interface would allow designers to develop their ideas as a marked-up sketch. The mark-up on the sketch would indicate critical elements and the relations between them [22].

Another critical area is explanation generation. We are currently developing novel explanation generation algorithms suited for interactive constraint satisfaction. Improved explanation generation capabilities will also provide us with a basis for providing feedback on comparisons between different design solutions that the designer wishes to consider.

Finally, the approach underlying the CAD technology described here has been evaluated in a number of industrial contexts. In particular, the approach has been evaluated

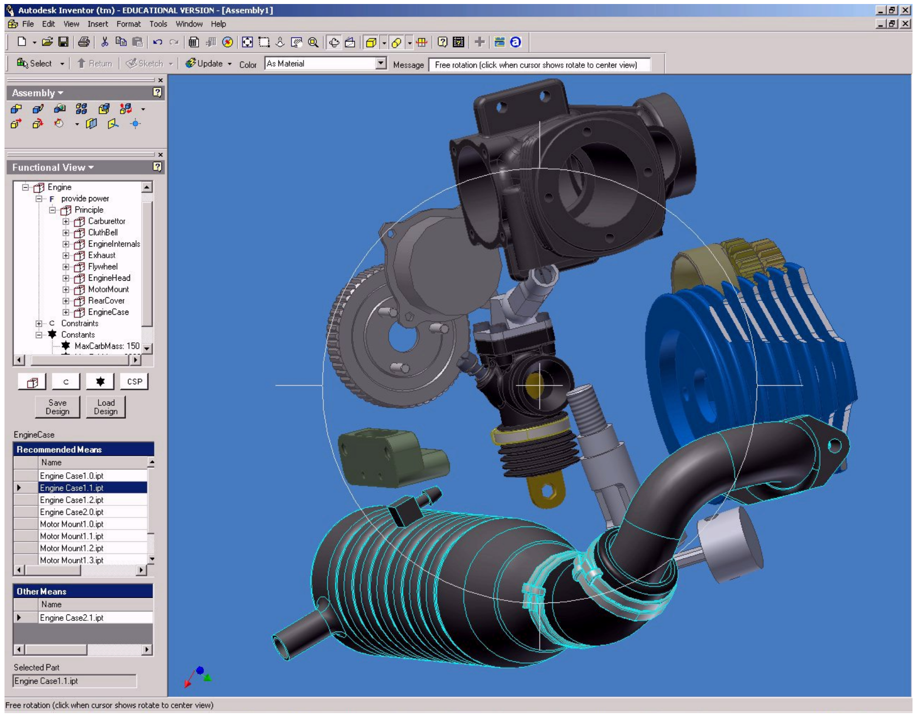

Fig. 10: Developing the scheme's part model in the host CAD system, Autodesk Inventor.

through several case-studies performed in conjunction with industry. Evaluations have been carried out in the fields of discrete electronic design, mechatronics and optical systems. For further details on how the technology has been used in industry, the reader is encouraged to refer to the literature available [14, 15]. In all cases designs of comparable sophistication to those developed by real-world designers and engineers using traditional methods were produced. Further evaluations will be undertaken using the prototypes developed as part of the work reported here.

6 Concluding Remarks

Engineering conceptual design can be defined as that phase of the product development process during which the designer takes a specification for a product to be designed and generates many broad solutions to it. It is well recognized that few computational tools exists that are capable of supporting the designer work through the conceptual phase of design. This paper presented a prototype constraint-based computer-aided design (CAD) technology that can be used to support designers working in the early stages of design. We presented the current status of our prototype and a detailed demonstration of its functionality. Finally, we described our current work on a number of additional features for the next prototype, which will be deployed in an industrial context.

References

1. L. Anthony, W.C. Regli, J.E. John, and S.V. Lombeyda. CUP: A computer-aided conceptual design environment for assembly modeling. Technical Report DU-MCS-01005, Department of Mathematics and Computer Science, Drexel University, June 2001.
2. D. Bahler, C. Dupont, and J. Bowen. An axiomatic approach that supports negotiated resolution of design conflicts in Concurrent Engineering. In *Artificial Intelligence in Design*, pages 363–379, 1994. Kluwer Academic Press, Netherlands.
3. C. Bessiere and J.-C. Regin. Arc consistency for general constraint networks: preliminary results. In *Proceedings IJCAI'97*, pages 398–404, 1997.
4. W.P. Birmingham and A. Ward. What is Concurrent Engineering? *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 9:67–68, 1995. Guest Editorial in a Special Issue on Concurrent Engineering.
5. J. Bowen and D. Bahler. Frames, quantification, perspectives and negotiation in constraint networks in life-cycle engineering. *Artificial Intelligence in Engineering*, 7:199–226, 1992.
6. J. Buur. *A Theoretical Approach to Mechatronics Design*. PhD thesis, Technical University of Denmark, Lyngby, 1990.
7. M.J. French. *Engineering Design: The Conceptual Stage*. Heinemann, London, 1971.
8. M.D. Gross, S.M. Ervin, J.A. Anderson, and A. Fleisher. Constraints: Knowledge representation in design. *Design Studies*, 9(3):133–143, July 1988.
9. D. Haroud, S. Boulanger, E. Gelle, and I. Smith. Management of conflict for preliminary engineering design tasks. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 9:313–323, 1995.
10. W. Hsu and B. Liu. Conceptual design: Issues and challenges. *Computer-Aided Design*, 32(14):849–850, 2000.
11. C. Lottaz, I.F.C. Smith, Y. Robert-Nicoud, and B.V. Faltings. Constraint-based support for negotiation in collaborative design. *Artificial Intelligence in Engineering*, 14:261–280, 2000.
12. C. Lottaz, R. Stalker, and I. Smith. Constraint solving and preference activation for interactive design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12:13–27, 1998.
13. S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *AAAI 90*, pages 25–32, July–August 1990.
14. B. O'Sullivan. *Constraint-Aided Conceptual Design*. PhD thesis, Department of Computer Science, University College Cork, Ireland, July 1999. (Also published by Professional Engineering Publishing, 2001, ISBN: 1-86058-335-0).
15. B. O'Sullivan. Interactive Constraint-Aided Conceptual Design. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 16(4):303–328, 2002.
16. G. Pahl and W. Beitz. *Engineering Design: A systematic approach*. Springer, 1995.
17. C. Petrie, H. Jeon, and M.R. Cutkosky. Combining constraint propagation and backtracking for distributed engineering. In *Workshop on Non-Standard Constraint Processing, ECAI 96*, pages 84–94, August 1996.
18. S. Y. Reddy, K. W. Fertig, and D. E. Smith. Constraint management methodology for conceptual design tradeoff studies. In *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, August 1996. Irvine, California.
19. D. Sabin and R. Weigel. Product configuration frameworks – a survey. *IEEE Intelligent Systems and their applications*, 13(4):42–49, July–August 1998.
20. D. Serrano. *Constraint Management in Conceptual Design*. PhD thesis, MIT, 1987.
21. S. Shimizu and M. Numao. Constraint-based design for 3D shapes. *Artificial Intelligence*, 91:51–69, 1997.
22. T. Stahovich, R. Davis, and H. Shrobe. Generating multiple new designs from a sketch. *Artificial Intelligence*, 104(1–2):211–264, 1998.