AUGMENTING DESIGN INTEGRATION AND COMMUNICATION USING IDIOM

Ian Smith

ABSTRACT: This paper discusses how a system developed for spatial composition using cases is able to improve design integration and communication. Layouts are built interactively by users rather than automatically generated as has been proposed by others. The design is incrementally parameterized as cases are added and therefore, case adaptation, user interpretation and model activation can occur at any stage. IDIOM supports designers through reducing constraint complexity and through managing design preferences, thereby restraining proposed solutions and further adaptation during subsequent project stages within **globally feasible design spaces**. *Since this system finds global solutions for constraints in an incremental manner, it is well suited for collaborative design and for increasing the integration of spatial design with other activities. Practical applications, currently under way, are demonstrating the advantages of IDIOM for communication and integration in construction.*

KEYWORDS: integration, design, spatial configuration, case-based reasoning, constraint satisfaction, model-based reasoning, preferences

1 INTRODUCTION

All complex engineering design tasks are carried out through cooperation between many partners. Better communication and integration in construction show much potential for reducing product costs, avoiding project delays and increasing product quality. Research related to improving communication and integration in construction has largely concentrated on data representation and communication only, leaving conflict management and consistency maintenance up to the users of the system. Detection and maintenance of globally consistent design solutions are often not addressed directly. When these issues are discussed, inadequate measures are proposed to provide support.

Unfortunately, recent practical experience with collaborative engineering has not been encouraging. Application of experimental communication networks to full-size tasks have resulted in *lower* quality of projects, *increased* project costs and *longer* project durations. It seems that since project partners can communicate information to every other partner at any time, they are less willing to follow traditional design flows and hierarchies and they are more likely to make important changes at late stages. Once they have communicated their design information, it becomes the responsibility of the other partners to ensure that their (the other partners') aspects of the project are consistent with the new project information. Partners become so *overloaded* with the quantity of design information that they can no longer ensure such consistency. Moreover, when parameters that are interconnected amongst requirements of several partners are involved, circular dependencies are created and this can lead to *divergent modifications as well as cycling* - even when globally compatible solutions exist.

Computer support for layout configuration has been studied for more than twenty years. Studies include techniques such as mathematical programming (Mitchell et al, 1976), optimization (Mitchell et al, 1976), space discretizations (Voss, 1994), genetic evolution (Gero and Schnier, 1995), graphs (Choi

and Flemming, 1995; Coulon, 1995), hierarchical generate and test (Flemming, 1988) natural language declarations (Fujii, 1995) and constraint satisfaction (Baykan and Fox, 1992; Medjdoub and Yannou, 1996; Tommelein, 1989). Rather than automate the configuration task, we have developed a system which supports designers as they compose designs *themselves* from parts of previous designs. Practising designers who collaborated within the scope of this study emphatically did *not* wish to have computer systems perform automatic layout generation.

This paper gives an evaluation of a system called IDIOM, which was recently developed for spatial composition tasks, for use in a collaborative engineering environment. Algorithms, which solve constraints rather than propagate parameter values, are described. Also, it is shown how models, preferences and constraint posting can be used to explore globally feasible solutions interactively. The goal of this paper is not to provide all details of IDIOM. A more complete presentation of IDIOM is given elsewhere, see Smith et al (1996).

2 IDIOM AND CASE-BASED DESIGN

Development goals for IDIOM did not specifically include improvement of design integration. We initially developed a system called Interactive Design using Intelligent Objects and Models (IDIOM) in order to study design interactivity, the use of preferences, and explicit domain modelling for case adaptation. Model-based adaptation was first proposed by Goel and Chandrasekaran (1989) for discrete variables. The term, IDIOM, was chosen because its meaning reflects an aim of this research. A dictionary definition (from Longman) for the word "Idiom" is

A phrase which means something different from the meaning of the separate words

This definition provides a useful analogy. We aim to support incremental composition of design cases while employing user interaction and domain models to include holistic considerations of groups of objects. Models are applied to designs several ways. They are activated when certain groups of objects are present in the design, they are used to interpret designs in certain contexts and they are incrementally introduced by the designer as the design is composed.

Our current research into case-based building design is motivated by two factors. The first factor is the observation that although building designers frequently reuse designs, they rarely wish to adapt whole building cases. Often, the cases which are most useful are spaces and collections of spaces (Schmitt, 1993). The second factor is that most design domains cannot be modelled completely due to a complex consideration of social, political and economic factors. As a result, it can be frustrating to designers when a system performs automatic design and proposes just one solution. A much better role for computer systems is to provide support for defining *allowable spaces of acceptable designs*. When exploring these spaces, designers are able to introduce *their interpretation* of what is not modelled through user interaction.

These two factors lead to the definition of an **intelligent object** that is used in this paper: an intelligent object is a *part of a successful design* which has been enriched by designers for each new design task through constraint posting, declaration of neighbourhood relationships, adaptation and model activation. Therefore, an object becomes intelligent at run-time. Such enrichment is used to accommodate additional objects during subsequent design stages. The notion of an intelligent object is not new, for example see Rigopoulos and Oppenheim (1992). An example of an intelligent object is a living room

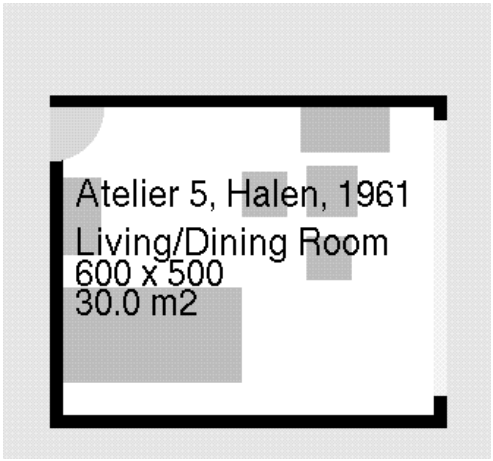

Figure 1: *An example of a case in IDIOM.*

taken from a design of a previously built apartment building. This living room becomes an intelligent object when i) the user interprets its new context by imposing conditions such as neighbourhood relationships and ii) when the user activates domain models to add additional constraints, such as the size of the living room needed for the number of inhabitants in the apartment. More detail of the models employed is provided in Section 2.2.

2.1 Cases in IDIOM

As mentioned earlier, IDIOM uses cases to build intelligent objects. Cases in IDIOM are parts of designs of constructed apartment buildings. Cases have been carefully selected by an architect for flexibility, compatibility and success as designs of parts of existing buildings. They are grouped into types such as living rooms, kitchens, bathrooms and bedrooms. They contain windows, furniture and doors. An example of a case is shown in Figure 1.

Grey rectangles within spaces represent furniture elements. The size of these rectangles include the size of the element plus additional space necessary for adequate use. For example, the size of a rectangle representing a dining room table includes an allowance for chairs as well as adequate room for sitting in them. Other elements shown in Figure 1 are the window in the right wall and the door on the left wall. The outer dimensions of the case as well as the positions of elements such as windows, doors and furniture are treated as variables. Sizes of elements within cases are fixed. All variables start with default values that correspond to their values in the original design. The origins of the case are described by the location of the building and the name of the architect.

2.2 Models in IDIOM

Models in IDIOM are causal mappings from structural parameters to behaviour related to individual objects (interpreted cases) and object groups. Behaviour is interpreted for a given context to correspond to a desired function. Therefore, model formulation follows the no function-in-structure principle (DeKleer and Brown, 1984; Gero, 1990). The definition of function, behaviour and structure follows Gero (1990).

We employ models to provide domain knowledge as configurations are composed. Models are abductively implemented through causal inversion (desired behavior to required structure). Since abduction is unreliable when a closed-world assumption is inaccurate, models in IDIOM are interactively activated, thereby providing one of several ways for the designer to introduce a problem-specific interpretation of the context.

3 INTEGRATION THROUGH CONSTRAINT SOLVING

Spatial composition of intelligent objects requires consideration of many interacting relationships between variables. Integration of several viewpoints is supported through incrementally solving relevant constraints, thus taking advantage of inter-relationships to reduce complexity.

Arrangements of intelligent objects and their elements such as doors, windows and pieces of furniture are defined by sets of constraints. Constraint sets have to be solved rapidly in order to allow interactive use, therefore we restrict these to linear and simple non-linear relationships. Relationships can be equalities or inequalities.

One of the most important aspects of the solver in IDIOM is its compatibility with interactive adaptation. When additional constraints are added, IDIOM finds a solution whilst maintaining positions and sizes in the current design wherever possible. Many algorithms in linear programming cannot do this. For example, those which employ pre-defined objective functions cannot dynamically add parametric values to the optimization criteria.

3.1 Sources of Constraints

There are three sources of constraints: the library of cases, the interpretation of the design by users and domain models. When a case is introduced into a design, all its associated constraints are added to the current set of constraints. Designers and other actors then add further constraints in order to interpret the case in its new environment. The most important constraint is the specification of the topology of the design, done by defining neighbourhood relations between objects. In addition, the user can specify constraints on the sizes, distances and alignments between objects and their elements. Before a new solution is calculated for the layout, constraints from active domain models are added to the current set of constraints. All constraints restrict values of continuous variables.

Equalities in the constraint set reduce the degrees of freedom of design spaces. This approach has been used in statistics (Krishnaiah and Kanal, 1982) and image recognition (Saund, 1989) and was proposed for case-based design by Faltings (1991). Subsequent development established that equalities can be used to reduce the number of variables occuring in inequalities (Hua, 1994). IDIOM thus uses Gauss-Jordan elimination to perform dimensionality reduction and to identify dependent and independent variables. In the inequalities, dependent variables are substituted by independent ones, thereby finding the matrix of coefficients of the equalities and inequalities.

For inequalities, IDIOM employs the Fourier-Motzkin elimination method. The procedure involves eliminating all variables one by one until a simple inequality-system with only one variable is found (Schrijver, 1986). Using intervals of possible values, it is easy to find a solution which is as near to the current solution as possible. The solver chooses a value for a variable by checking its interval of

possible values. If the current value of the variable is within the interval the solver will use this value. If the value is outside it will be set to the nearest interval boundary.

3.2 Preference Activation

Constraints in IDIOM may be fixed or preferred, hereafter referred to respectively as fixed constraints and preferences. Fixed constraints must be fulfilled while preferences may be deactivated if they are in conflict with other preferences or fixed constraints. Preferences are reactivated when possible. The priority of a preference can be defined and preferences may have equal priority. IDIOM fulfils all fixed constraints and as many preferences as possible using the following heuristics :

- A preference that conflicts with fixed constraints is deactivated
- If two preferences with different priorities conflict, the higher priority preference is activated
- If two preferences with the same priority conflict, IDIOM activates the preference which conflicts with fewer lower priority preferences
- IDIOM re-activates preferences whenever possible

Preferences are divided into groups of equal priority and activated in order of importance. For example, six preferences are divided into three groups according to priority. The most important group g_1 contains p_1, p_2 and p_3, the second group g_2 contains p_4 and p_5 and the least important group g_3 contains p_6.

The activation of preferences starts with none activated; as many preferences as possible are activated in the first group through checking feasibility with all fixed constraints. This is performed incrementally for each preference. Several feasible combinations of preferences may have the maximum number of preferences activated and therefore these are stored into a list of solutions. In this example, two preferences out of g_1 can be activated and the following combinations are possible: $\{p_1, p_3\}$ and $\{p_2, p_3\}$. Then L, the list of solutions after treatment of g_1, is :

$$L = (\{p_1, p_3\}, \{p_2, p_3\})$$

The activation of preferences then sequentially considers all entries in the list with additions from g_2, and stores all combinations which have the maximum number of preferences activated. Thus the combination $\{p_1, p_3\}$ is considered first and IDIOM finds that only p_4 can be added. Then preference activation treats the combination $\{p_2, p_3\}$ and finds for instance, that only p_5 can be activated together with this second combination. Thus two solutions are found and a new list is created :

$$L = (\{p_1, p_3, p_4\}, \{p_2, p_3, p_5\})$$

After treating all preference-groups in this manner, preference activation terminates with a list of feasible combinations which contain as many important preferences as possible. One of these is then used to recalculate the new values of the design's parameters and for subsequent adaptation. For example the preference in g_3 can be added with the second combination in L, but not with the first combination. The final list contains one combination of feasible preferences which is used in further calculations.

$$L = (\{p_2, p_3, p_5, p_6\})$$

4 IMPLEMENTATION AND SCENARIO OF USE

IDIOM is implemented in C and C++ with OpenGL and Motif as the user interface platform. The following is an example of a possible design scenario using the system (each step performed by the user):

1. Define the dimensions of the site where the layout must be placed
2. Choose a case from the case browser and place it into the site. At this point, constraints contained in the case and those activated by models are added to the constraint set
3. Define neighbourhood relationships with adjacent objects. This action automatically adds more constraints to the constraint set
4. Where needed, post additional constraints
5. Request solution. Here the system calculates the feasible solution space through conflict resolution with preferences and dimensionality reduction and selects a solution that involves minimal changes to the case and to the current design
6. Interactively adapt positions of walls, furniture, windows and doors to obtain configuration required
7. If design is incomplete, return to step two
8. As design evolves through detailing and construction, update decisions in order to ensure that new requirements do not conflict with constraints posted during initial layout composition.

The last step is the most important for communication and integration. It is not suprising therefore, that this aspect has created so much interest amongst designers who have agreed to contribute to further development of IDIOM. IDIOM provides an opportunity to create a link between constraints considered in the design office and inevitable downstream changes.

The screens shown in Figure 2 refer to step 2 on the left and step 5 on the right. On the left, a double bedroom is being added to the design. After user interpretation, in this case specifying that the hall should share the length of the right wall through declaration of a neighbourhood relationship, the solution proposed is shown on the right. Note that the vertical dimensions of both the hall and the bedroom have changed.

Once a layout has been composed, user may change positions of walls and elements within objects while maintaining integrity with respect to design requirements. This is carried out through clicking on a wall or element. The results of the dimensionality reduction are used to calculate the range of adaptation possible. Arrows show the range of globally consistent solutions. The screens in Figure 3 show that moving a wall may change other dimensions that are linked in the parameterization. Here, the living room has been constrained to have the same length to width ratio, the bathroom dimensions have been fixed and the kitchen wall is required to share the whole right wall of the living room. Therefore, moving the living room wall results in a reduction in size of the single room at the top.

Figure 2: *Adding a case to a design (left) and solution proposed after user interpretation and resolution with relevant constraints (right).*

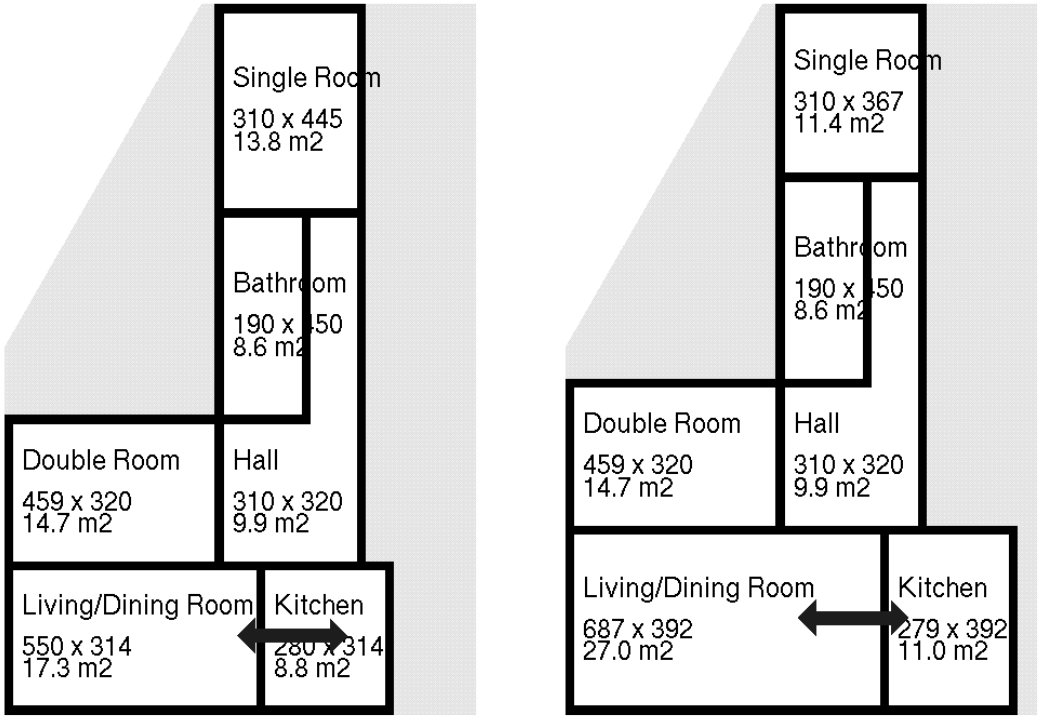

Figure 3: *The figure illustrates Step 6, interactive adaptation. The user clicks on the right wall of the living/dining room (left) and drags it to the desired position (right). Note the changes to dimensions of the single room at the top.*

5 APPLICATION EXPERIENCES

Designers in industry have been very fast to recognize IDIOM's potential for improving design communication and integration. Several people can contribute to IDIOM's constraint set without loss of functionality. More importantly, IDIOM gives them the possibility to attach consistent design requirements to designs in way that they are able to stay with the design up to and including construction stages. Current design applications under study are residential building design and exhibition floor layout. IDIOM is also under study for an application in construction site layout. Again, the general contractor who is responsible for site layout is interested in IDIOM mostly because the system is capable of maintaining globally consistent solutions throughout the duration of construction. Support initial configuration task (the original goal of IDIOM) is seen almost as a nice extra.

6 RELATED WORK

6.1 Design Integration and Communication

Earlier work includes IBDE (Fenves et al, 1988) and DICE (Sriram et al, 1990). This work concentrated on acheiving local consistency through creation of blackboard architectures. More recent work has focused on communication systems where designers perform most of the negotiation and conflict management required to complete globally consistent designs. Work at Boeing (Klein, 1995) proposed rules for achieving global consistency. Other work includes Oh and Sharpe (1995), Saad and Maher (1995), Fructer (1996), Petrie (1995), Pena-Mora et al (1995), the SHADE system at Lockeed (McGuire et al, 1993), SHARE (Toye et al, 1995) as well as European projects such as COMBINE, ATLAS, CIM-Steel and COMBI (Scherer, 1995). Most of this work concentrates on improving the communication of design information only, leaving conflict management and consistency maintenance up to the users of the system. When support for conflict management is provided (Klien, 1995; Oh and Sharpe, 1995) global consistency between all design viewpoints is not maintained explicitly. IDIOM offers a direct and reliable method for calculating feasible solution spaces through solving all relationships together. IDIOM is limited however to equalities and inequalities between continuous variables that are linear or can be approximated by sets of linear relationships (Smith et al, 1996).

6.2 Spatial Configuration and Case-Based Design

The system most closely related to IDIOM is CADRE (Dave et al, 1994; Hua, 1994; Hua et al, 1992). Similarities include i) the use of dimensionality reduction and run-time parameterization to simplify adaptation and ii) certain aspects of user interaction, such as the use of arrows for defining feasible modifications. IDIOM differs from CADRE in the following ways: i) IDIOM employs intelligent objects to compose topological configurations where the CADRE implementation combines predefined configurations, ii) IDIOM accommodates preference constraints whereas in CADRE, all constraints are fixed, iii) in IDIOM, elements within spaces, such as furniture, doors and windows are included in the parameterization whereas in CADRE, only spaces and structural elements are included, iv) IDIOM employs a more reliable algorithm for accommodation of inequalities during case combination (Smith et al, 1996), v) IDIOM employs explicitly defined domain models that are activated by the user whereas in CADRE domain knowledge was loaded into the system at the beginning and finally, vi) the

opportunities for interactivity in IDIOM correspond more closely to the needs of building designers who were interviewed than in CADRE. Perhaps the most important difference between IDIOM and CADRE is that in IDIOM, the topology is determined interactively by the user, thereby avoiding difficulties of complexity experienced with CADRE when generating topologies.

The FABEL project, coordinated by GMD, St. Augustin (Baktari et al, 1993) focuses upon the application of case based design to heating and ventilating configuration of buildings. Although most of the effort in this project has been concerned with case indexing, recent work includes a study of three adaptation methods (Boerner, 1995). FABEL uses fixed grids to model spatial information and does not perform case-combination. Other work includes the SEED project (Flemming, 1994) where large numbers of cases are stored and indexed for retrieval using functional units. Although a case editor is a available for adaptation, no other computational support is reported. Our approach is different to these two systems due to our capabilities to combine complex objects through run-time parameterizations that include design preferences, constraints from user-activated models and other opportunities for user interpretation.

An extension to CADSYN (Zhang and Maher, 1993) employs constraint satisfaction techniques for verification and repair of adapted designs. CADSYN ensures local consistency between constraints, thereby limiting its effectiveness to constraint networks where risks of divergence, looping and empty solution spaces are low. Our experience with geometric design has revealed that relevant constraint networks are highly interdependent and therefore, local consistency approaches are unreliable.

WRIGHT is another constraint based system created for layout synthesis (Baykan and Fox, 1992). Layouts are automatically generated and local consistency is achieved through use of the Waltz algorithm, thus risking cycles and divergence. In a comparison with methods based on hierarchical generate and test (Flemming et al, 1992), it was concluded that constraint propagation techniques are more efficient under certain circumstances. Since WRIGHT performs monotonic search, soft constraints are never reactivated if weakened. IDIOM differs from WRIGHT in the following ways : i) layouts are not generated in IDIOM – the user defines topology incrementally, ii) IDIOM does not propagate constraints but *solves* them, and together with the Fourier-Motzkin algorithm, identifies globally feasible solutions without running the risk of propagation cycles and iii) IDIOM may demonstrate non-monotonic behaviour as cases are added since preferences may be reactivated.

Work on layouts is being performed by Giretti et al (1994). They report on a CBD system for architecture that supports graphical interaction. Their "theories" are similar to the models in IDIOM and "scenes" are analogous to groups of intelligent objects. However, run-time parameterization and subsequent dimensionality reduction is not performed. Therefore, performance problems would be expected for designs of realistic size. In addition, it is not clear whether local or global consistency is achieved during constraint solving. Finally, grids are used to adapt previous designs and topological adaptation is performed automatically. These functionalities were avoided during development of IDIOM because designers who were interviewed thought they would hinder rather than help layout design.

7 CONCLUSIONS

IDIOM provides a useful framework for integrating activities related to spatial composition. Through solving constraints contained in cases, those generated during user interaction and those obtained from model activation, users are able to explore a range of design solutions within globally consistent design spaces. This exploration is further enhanced by the accommodation of preferences in constraint sets

and the opportunity to alter preference priorities interactively. Algorithms are sufficiently fast and reliable to support exploration in an interactive manner. Reactions from designers indicate that IDIOM provides a good mix of computation and user support for spatial configuration. Since constraints are solved rather than subjected to value propagation, constraint circularity does not restrict maintenance of globally consistent solutions in the system. This facility shows much potential for the AEC industry.

ACKNOWLEDGMENTS

The funding for this project was provided by the Swiss Priority Programme in Computer Science (SPP-IF). This project was performed in collaboration with CAAD, Federal Institute of Technology, Zurich. Beginning ideas related to intelligent objects arose during discussions with Boi Faltings and Gerhard Schmitt and Boi Faltings initially proposed an investigation of model-based case adaptation for this work. The author would like to thank Ruth Stalker, Claudio Lottaz, Nathanea Elte and David Kurmann for their collaboration and work implementing IDIOM as well as our practising architects : Geninasca - Delfortrie Architects, Neuchatel; Atelier d'Architecture, Lausanne; and Archilab, Lausanne for providing comments and for helping with testing and validation.

REFERENCES

Bakhtari S. et al, (1993) "EWCBR93 : Contributions of FABEL" Fabel Report No. 17, GMD, Sankt Augustin.

Baykan, C.A. and Fox, M.S. (1992) "WRIGHT: A constraint based spatial layout system" AI in Engineering Design, 1, Academic Press, pp 395-432.

Börner, K. (1995) "Modules for design support", FABLE Report No. 35, GMD.

Choi, B. and Flemming, U. (1995) "Adaptation of a layout design system to a new domain" CAAD Futures'95, Singapore.

Coulon, C-H. (1995) "Automatic indexing, retrieval and reuse of topologies in architectural layouts" CAAD Futures'95, Singapore.

Dave, B., G. Schmitt, B. Faltings and I. Smith (1994) "Case-based design in architecture" AI in Design '94, J.S. Gero and F. Sudweeks (eds.) Kluwer, pp 145-162.

De Kleer, J. and Brown, J.S. (1984) "A qualitative physics based on confluences", Artificial Intelligence, 24.

Faltings, B. (1991) "Case based representation of architectural design knowledge" Computational Intelligence, 2.

Fenves, S., Flemming, U., Hendrickson, C., Maher, M.L., and Schmitt, G. (1988) "An integrated software environment for building design and construction", 5th International Conference on Computing in Civil Engineering, ASCE.

Flemming U. (1994) "Case-based design in the SEED System", 1st Computing Congress, American Society of Civil Engineers, Washington.

Flemming U. Baykan C.A. Coyne R.F. and Fox M.S. (1992) “Hierarchical generate and test vs constraint-directed search” AI Design’92, J.S.Gero(ed) Kluwer, 817-838.

Flemming U., R. Coyne, T. Glavin and M. Rychener (1988) “A Generative Expert System for the Design of Building Layouts”, AI in Engineering, Elsevier, 445-464.

Fujii, H. (1995) “Incorporation of natural language processing and a generative system” CAAD Futures’95, Singapore.

Gero, J.S. (1990)“Design prototypes: a knowledge representation schema for design”, AI Magazine, 11(4), 26-36.

Gero, J. S. and Schnier, T. (1995) “Evolving representations of design cases and their use in creative design”, Preprints Computational Models of Creative Design (to appear), 1995. http://www.arch.su.edu.au/~john

Giretti, A., Spalazzi, L. and Lemma, M. (1994) “A.S.A. An interactive assistant to architectural design” AI in Design’94, J.S.Gero and F.Sudweeks(eds) Kluwer, 93-108.

Goel, A.K. and Chandrasekaran, B. (1989) “Use of device models in adaptation of design cases” DARPA CBR Workshop, pp100-109.

Hua, K. (1994) “Case-based design of geometric structures” Thesis No 1270, Swiss Federal Institute of Technology, Lausanne.

Hua K., Smith, I., Faltings, B., Shih, S., and Schmitt, G. (1992) “Adaptation of Spatial Design Cases” AI in Design’92, J.S. Gero (ed.) Kluwer, Dordrecht, NL.

Klein, M. (1995) ”Conflict management as part of an integrated exception handling approach”, Artificial Intelligence for Engineering Design Analysis and Manufacturing, 9 (4).

Krishnaiah, L. and Kanal, P., (1982) Handbook on Statistics, Volume 2, North-Holland, Amsterdam.

McGuire, J.G., Kuokka, D.R., Weber, J.C., Tenenbaum, J.M., Gruber, T.R. and Olsen, G.R. (1993)“SHADE: Technology for Knowledge-Based Collaborative Engineering”, Concurrent Engineering: Research & Applications, Volume 1, Number 3.

Medjdoub, B. and Yannou, B. (1996) “Towards a new generation of architectural CAD softwares”, accepted for ITCSED-96, Glasgow, Scotland.

Mitchell, W.J. Steadman, J.P. and Ligget, R.S. (1976)“Synthesis and optimisation of small rectangular floor plans” Environment and Planning B, 3, 37-70.

Rigopoulos, D.R. and Oppenheim, I.J. (1992) “Intelligent objects for synthesis of structural systems” Journal of Computing in Civil Engineering, ASCE, 6, 266-281.

Saund, E. (1989) “Dimensionality reduction using connectionist networks”, IEEE trans. PAMI, 11, 304-331.

Scherer, R. (ed.) (1995) “Product and process modelling in the building industry”, Balkema, Rotterdam, 606p

Schmitt, G. (1993) “Design reasoning with cases and intelligent objects” International Association of Bridge and Structural Engineering, Report 68, pp 77-87

Schrijver, A. (1986) “Theory of linear and integer programming” John Wiley & Sons, Chichester.

Smith, I., Stalker, R., and Lottaz C. (1996) ”Creating design objects from cases for interactive spatial composition” 4th International Conference on AI in Design, Stanford, Proceedings to be published by Kluwer.

Sriram, D. Lodger, R. Wong, A. and Ahmed, S. (1990) ”A case study in computer-aided cooperative product development” Technical Report IESL-90-01, Intelligent Engineering Systems Laboratory, MIT.

Tommelein, I.D. (1989) “SightPlan - An expert system for designing construction site layouts”, PhD Thesis, Stanford University.

Toye, G., Cutkosky M.R. and Leifer, L. (1995) ”SHARE: A Methodology and Environment for Collaborative Product Development” Post-Proceedings, IEEE Infrastructure for Collaborative Enterprises.

Voss A. (1994) “The need for knowledge acquisition in case-based reasoning - some experiences from an architectural domain”, 11th ECAI, John Wiley, pp463-467.

Zhang, D.M. and Maher, M.L. (1993) “Using CBR for the synthesis of structural systems” Inter. Assoc. for Bridge and Structural Engineering, Report 68, 143-152.