INCREASING UNDERSTANDING DURING COLLABORATION THROUGH ADVANCED REPRESENTATIONS

SUBMITTED: July 1999
REVISED: January 2000
PUBLISHED: February 2000 at http://itcon.org/2000/1/
EDITOR: Z. Turk

Claudio Lottaz
Artificial Intelligence Laboratory, Swiss Federal Institute of Technology Lausanne, Switzerland
email:Claudio.Lottaz@epfl.ch http://liawww.epfl.ch/~lottaz/

Rudi Stouffs, Ph.D.
Architecture and CAAD, Swiss Federal Institute of Technology Zurich, Switzerland
email:stouffs@arch.ethz.ch http://caad.arch.ethz.ch/~stouffs/

Ian Smith, Prof.
Structural Engineering and Mechanics, Swiss Federal Institute of Technology Lausanne, Switzerland
email:Ian.Smith@epfl.ch http://imacwww.epfl.ch/team/smith/ian_smith.html

SUMMARY: *Efforts to provide support for collaborative work in the AEC industry have resulted in systems that offer various levels of assistance. Although some systems support information transfer in a wide range of formats, they offer little in terms of decision support such as conflict management and negotiation. Other systems provide more decision support but require strict formats for information input and transfer. Nearly all current proposals offer very limited facilities for viewing information. The objective of this paper is to present an environment which has been specifically designed for multiple ways to represent and manipulate information. Several representations, when coupled with appropriate visualization techniques, lead to opportunities for increasing understanding of AEC project characteristics. More specifically, when a numerical constraint solver (SpaceSolver) is integrated within a document-centric collaboration environment (ICC), synergies between information exchange and solution space exploration contribute very positively to the quality of projects. In particular, the ICC environment provides a framework for representing and visualizing information structures that are created during collaboration. Conceptually, an information architecture and visualization techniques to support the virtual AEC enterprise are emphasized. A plug-in architecture allows for the addition of process-specific functionality. The constraint solver SpaceSolver presents a complementary collaborative approach, with strict semantics to support decision making and conflict management. The use of solution spaces during collaborative negotiation avoids premature decisions in the design process, allows detection of conflicting project requirements at early stages of the project, and increases the designers' understanding of hidden relations between design parameters. Together, the ICC environment supports the management of an information space that, when linked to a constraint satisfaction problem, can explain important restrictions and decisions for an effective negotiation. The combination of a flexible framework with more rigid modules, such as constraint solvers, provides a useful compromise and, thus, comprehensive support for a range of AEC projects. Two recently completed construction projects are used to validate the approach.*

KEYWORDS: *collaboration, negotiation, electronic document management, information architecture, information visualization, feedback.*

1. INTRODUCTION

Most engineering tasks require collaboration between many partners. Collaboration tasks are complicated by factors such as time and data losses during information exchange, misunderstandings because of ill-defined information, and iterative negotiation when subtask solutions conflict. Moreover, changes in context, costs, requirements, deadlines, etc., require constant renegotiation of issues that can modify important project characteristics. During this negotiation it is entirely the responsibility of the collaborators to ensure consistency, to consider all important alternatives, and to inform the partners of important justifications for decisions.

Working with our AEC industry partners in the scope of the project “A tool set for the virtual AEC industry” (Schmitt et al, 1999) has demonstrated the need for software tools that would enable these and other companies to interact and exchange information with several partners without the need for time-consuming physical meetings, complemented by the wish to have up-to-date, secure and consistent information on project characteristics. At the same time, many companies do not consider the AEC community in general to be ready to embrace such technologies on a large scale. These companies are particularly worried that if all except one partner in a team have access to and experience with new technology, the one without it may slow and possibly break the information flow. This company might be crucial for the team for other reasons, so that a cycle of partial implementation – partial success starts.

Even when adopting paperless technology, much trouble may be caused during negotiation among collaborators due to the practice of participants suggesting only single solutions. Collaborators do not usually gather the necessary information in order to determine whether the problem at hand actually has a consistent solution. Furthermore, if a solution exists, ranges of possible values for variables are not determined. The use of constraints as mathematical equations on continuous variables for specifying requirements can assist in deriving such information. When collaborators specify their requirements in this way, computational tools can approximate the space of potential solutions, thus providing means to detect conflicts and to guide negotiation.

We suggest the combination of both approaches for communication and information representation. The ICC (Information, Communication, and Collaboration) environment provides support for general information sharing in the context of a collaborative building project. *Space*Solver is a communication platform based on the strict semantics of variables and constraints as information entities. Together, these enable the same design information to be represented and exchanged both in a strict mathematical way as well as in a free-form way. *Space*Solver provides specific support for negotiation and decision making using constraint satisfaction techniques. The ICC environment allows this information to be extended with explanatory and other types of information, and contains additional tools for visualizing information structures that are created during collaboration.

In this paper, we present an information architecture and visualization techniques to support the virtual AEC enterprise, and their implementation in the ICC environment, in section 3. Details on the use of constraints during collaboration and the role of *Space*Solver in this use are given in section 4. The combination of the two platforms is described in section 5.

2. BACKGROUND

The AEC industry has a rich history of collaboration, even within a context of fierce competition. In a struggle to become more agile (Preiss et al, 1997, Herbert, 1997), well-defined and understood control hierarchies and relationships make place for more ad hoc and intricate collaborative processes that are not as easily planned and controlled. Building collaborations are special in that both the project and the team, and as a result the processes, are potentially unique from project to project (Buckley et al, 1998). This requires an increasingly networked thinking that brings partners to closer interaction but, without appropriate computational support, impedes the ease of overview and understanding. The use of electronic data exchange in the AEC industry is sparse and, where available, localized around specific interactions (Almeida et al, 1998). Efforts to organize and support electronic exchange are hampered by the fragmented nature of the industry and the heterogeneity introduced by the fast technological evolution.

Most recent research efforts focus on long-term advances in product and process modeling (Almeida et al, 1998, Anumba et al, 1998, Buckley et al, 1998, Scherer, 1998). Even when development support is available from the industry (Wix, 1997, Beetz et al, 1998), the challenge remains to convince all participants in a building project to adopt the same technology. In the fragmented AEC industry, e-mail and access to the Web generally constitute the least common denominator for electronic exchange. This has resulted in a generation of Web applications for design project management that provide facilities for organizing, viewing, and redlining drawings and images (Roe, 1999). Methodologies of viewing shared documents in the form of hierarchies, lists, hyperlinked documents, or tables increasingly illustrate the limitations of such presentations. Instead, an advanced representation that captures the information structures built during collaboration, combined with appropriate visualizations of these structures, can empower the partners in the analysis and understanding of the collaborative processes and increase their effectiveness during collaboration.

Although communication and collaboration tools recently became more attractive with the success of the Internet, little work proposes support for maintaining project consistency. Many projects, e.g. (Cutkosky and Tennenbaum, 1996, Divita et al, 1998, Fruchter, 1996, Roddis, 1998, Rezgui et al, 1998), contain proposals for communication and information management facilities. However, inconsistent construction projects are costly and therefore, there has been research into explicit computer support for negotiation and conflict mitigation. Beginning with work into design rationale (Peña-Mora et al, 1995), Peña-Mora has recently proposed a combination of negotiation and game theory to support negotiation between partners (Peña-Mora, 1998, Peña-Mora and Wang, 1998). Ndumu and Tah (1998) suggest a computational market model to resolve conflicts. Mokhtar et al (1998) focus on change management to provide an information model that assists in planning and scheduling design changes. None of this work proposes constraint solving or solution spaces for negotiation support.

When constraint solving is used, only local consistency is usually assured. Bahler et al (Bahler et al, 1995, Bowen and Bahler, 1993) proposed a design advice tool that uses constraints to support negotiation and conflict resolution. An exception handling approach studied by Klein (1997) also uses local methods for enhancing consistency. Realizing the need for constraint solving in the Redux system, Petrie has enhanced the framework to include a constraint manager that would plug into remote solvers (Petrie et al, 1997). Khedro and Genesereth (1994) have developed a progressive negotiation strategy for conflict resolution where locally consistent solutions are used to converge on a global consistency. However in these studies, no explicit use of solution spaces has been found for constraints expressed in terms of continuous variables.

3. AN INFORMATION ARCHITECTURE FOR THE AEC INDUSTRY

The ICC system employs a Web-based environment to share and manage information in the context of a collaborative building project (Stouffs et al, 1998). It serves as a framework for the development and dissemination of tools that can be used by both a single partner and the entire team. Of particular interest is the development of tools to support collaborative processes and the visualization of information structures that are built during collaboration. Use of these tools leads to a better understanding of collaborative activities.

3.1 Aspects of the information architecture

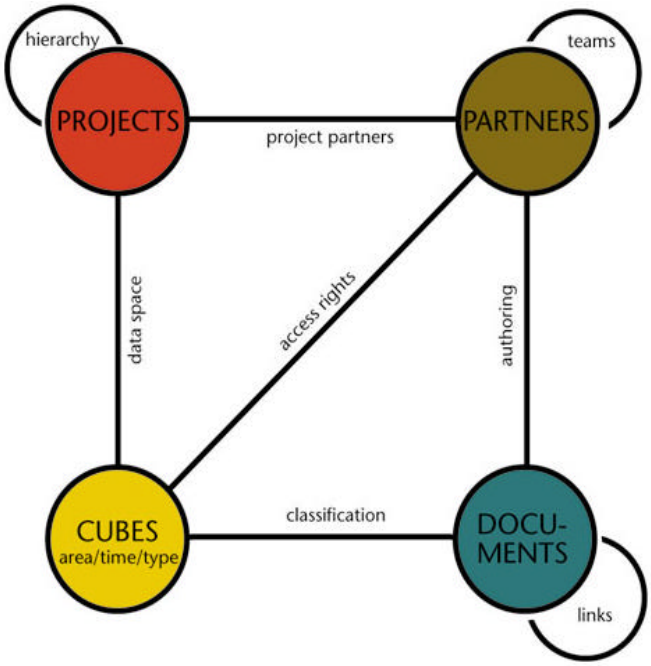

FIG. 1: Schematic view of the data structures underlying the information architecture.

A few information aspects are indispensable for defining, building, and visualizing information structures. These are the information entities that provide the resources for all activities, a project organization that assists in

managing these entities, authoring information that attributes credits and assigns responsibilities, and relationships that embed the collaborative structure. Fig. 1 presents a schematic view of the data structures that support these aspects.

Fig. 2 illustrates these information aspects in a view of the prototype interface. The top frame enables the search and retrieval of information entities through access to the project organization. Entity sets are presented in the left frame; the presentation hierarchy is derived from entity relationships. Detailed information of an entity, including authoring information, is shown in the right frame. The frame below the two main frames contains an alternative presentation of the same entity set. Finally, the bottom left frame provides iconized access to environment plug-ins.

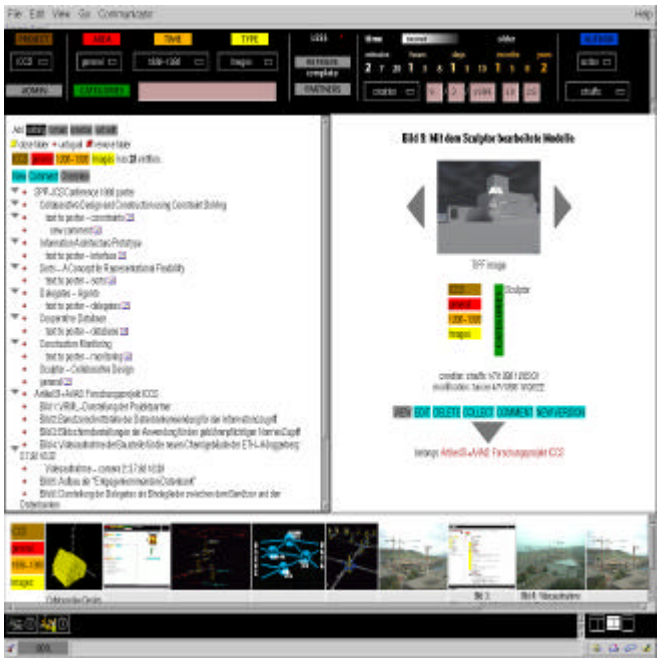

FIG. 2: View of the ICC prototype interface. Bige Tunçer and Rudi Stouffs.

3.1.1 Information entities

A document modeling approach, where the information entities in the collaborative structure are defined by the documents submitted by the participants, allows for maximal flexibility in specifying the information space. Each entity corresponds to a single document (or text) and its related information, including authoring information, a categorization with labels, and user-defined attributes. The formats for these documents are defined by the tools and applications that the participants adopt. Their exact formats do not necessarily have to be known to the environment; additional support for different formats may be provided by browser plug-ins or environment extensions. One such extension constitutes the link to *Space*Solver. Through this link, the ICC environment provides for the management of design constraints and variables with related information. Whilst these constraints and variables are considered textual information entities by the environment, they correspond one-to-one to design entities as recognized by the constraint solver (see section 5).

3.1.2 Organization

An appropriate organization of the information entities assists participants when searching, browsing, and managing project information. The ICC system uses a classification of the information entities within a project according to three dimensions, similar to the ZIP cube (Arb et al, 1997). Whereas the indices of the ZIP cube are exactly defined (according to established practices in the Swiss AEC industry) the specification of these dimensions in this environment is left to the project team in order to reflect on the specifics of the project and the anticipated processes. Documents can be submitted, selected, and visualized by project and with respect to this three-dimensional structure. A VRML visualization of this organizational structure provides a navigable overview of the project organization (Fig. 3). Component cubes are sized with respect to entity count, and

highlighted in the structure according to selected criteria, e.g., whether there are new documents or entities waiting attention. Upon selecting a cube, an overview of relevant entities is presented in the interface.

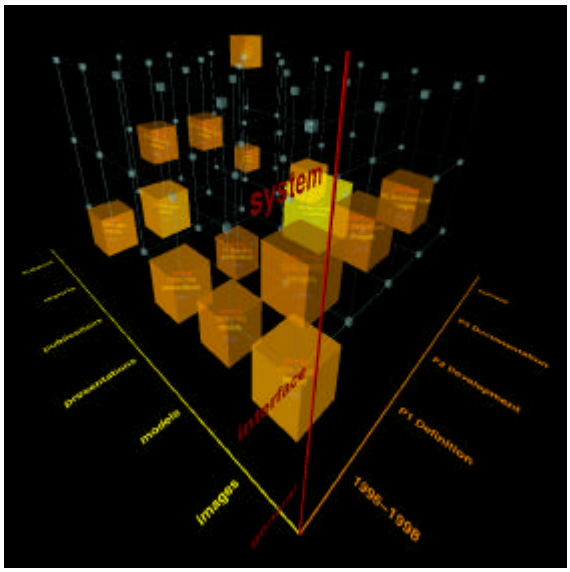

FIG. 3: A 3-dimensional visualization of the organizational structure of a project. Bige Tunçer.

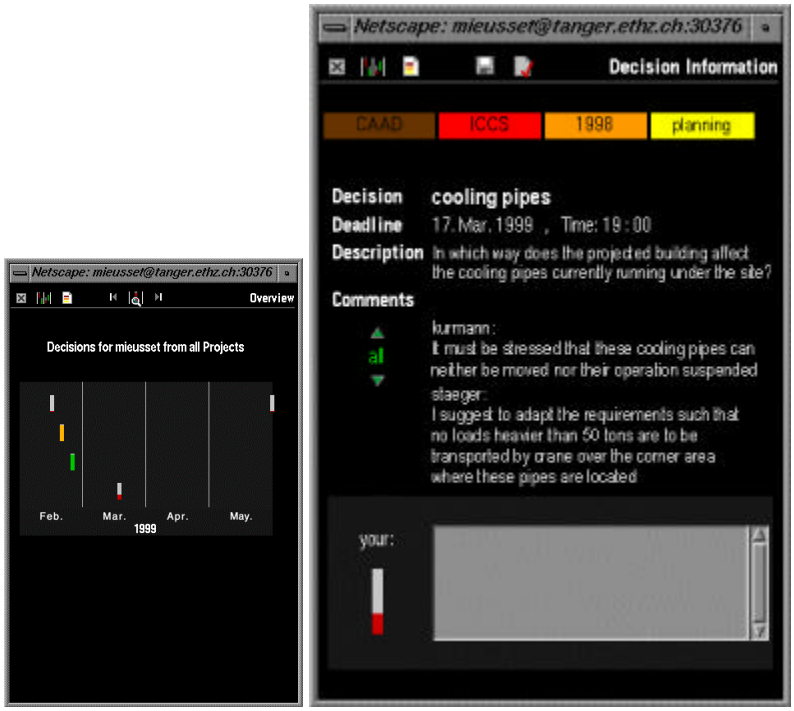

FIG. 4: Decision support plug-in: overview (left) and view of a discussion status (right). Kuk Hwan Mieusset and Benjamin Stäger.

3.1.3 Authoring information

For a collaboration to be effective, it is important that the participants are known and recognized for their part in the collaborative process and resulting information space, both in terms of credit and responsibility. Registered project partners are authenticated by the environment. Authoring information is automatically recorded and assigned to a document, and collaborative authors can be ascribed to individual documents. Authoring

information credits individual contributions and affords feedback on the role of a participant in the collaborative process. Ascribing collaborative authors to a document assigns both access rights and responsibilities. An environment plug-in for decision support (Fig. 4) uses this ascription facility to specify and manage decision teams, and elicit discussions. It presents the user with an overview of all current discussions that involve the user, highlighting those that await a decision or standpoint from the user. Participants' responses are collected and attached to the discussion entity and managed by the plug-in in order to determine a discussion's progress status.

3.1.4 Relationships

The information structure resulting from a collaborative process is visualized from the information entities and the corresponding links between these entities, as created by the participants in the process. Links allow the user to express relationships, browse the data space, and can assist in interpreting the information space. A measure of density, as expressed by the number of links that connect to an entity, especially in combination with time information, can also lead to the recognition of activity centers. Some types of links are self-evident and are maintained by the environment. These allow one to group entities, e.g., a set of images with the documents these appear in, specify threads of messages or attach messages or comments to other information entities, and specify versioning sequences in collaborative work. Other links are left to the discretion of the users, or are additionally supported by extensions to the environment.

3.2 Visualization of the information structures

Effective visualizations should enable a more effective and efficient collaboration among the participants through a visual analysis of the information structure and the underlying collaborative processes. In particular, these can serve to guide the user to zones or nodes of interest, highlight problems or issues that need consideration, determine activity centers, and illustrate complex processes.

In order to achieve such presentations, entity attributes such as position, size, shape, and color can be used to distinguish components with respect to any aspect that is deemed important. Additionally, using grades of size, intensity, or transparency, entities and relationships can be emphasized and de-emphasized (or even omitted) in order to accentuate certain aspects. In order to alter the character and the focus of a visualization one can vary the scope (from a single entity to the entire structure), the interactivity (whether solely for the purpose of browsing and selection or augmented with all activities associated to an entity or relationship), and the presentation technique (from a static, pure-HTML page to a dynamic, four-dimensional visualization including time).

3.2.1 Time and authoring

Time and authoring information can illustrate the effect of individual actions and allow for a history of changes. The latter is especially important when dealing with collaborative documents. Fig. 5 shows an example from *Phase(x)* (Hirschberg and Wenz, 1997), an entry level CAAD course at the Chair for Architecture and CAAD (ETH Zurich), with an emphasis on collaborative authorship: an organization into different learning phases allows the results of one phase and one author to be taken as the starting point for the work in the next phase by a different author.

An overview (top) of the students' exercises and their relationships organized by the time of submission provides testimony to periods of increased or decreased activity, and hints at time dependencies between exercises in consecutive phases. The X-axis (horizontal) specifies the time of submission, the Y-axis lists the authors. The height of the solids corresponds to their respective rating by the authors' community; contributions near the end of the semester were not rated. Another view (bottom) characterizes each contribution both by the duration between start and submission of the exercise (length of the solid) and the number of dependent or derived exercises (height of the solid), enabling an interpretation of the effect of the former on the latter. The location of each contribution on the X-axis (horizontal) reflects on the phase, on the Y-axis on the genealogy (the relationships between exercises in consecutive phases). The color scheme in Fig. 5 highlights the threads of exercises that build upon an initial contribution, presenting feedback on the role of an individual exercise in the collaborative process.

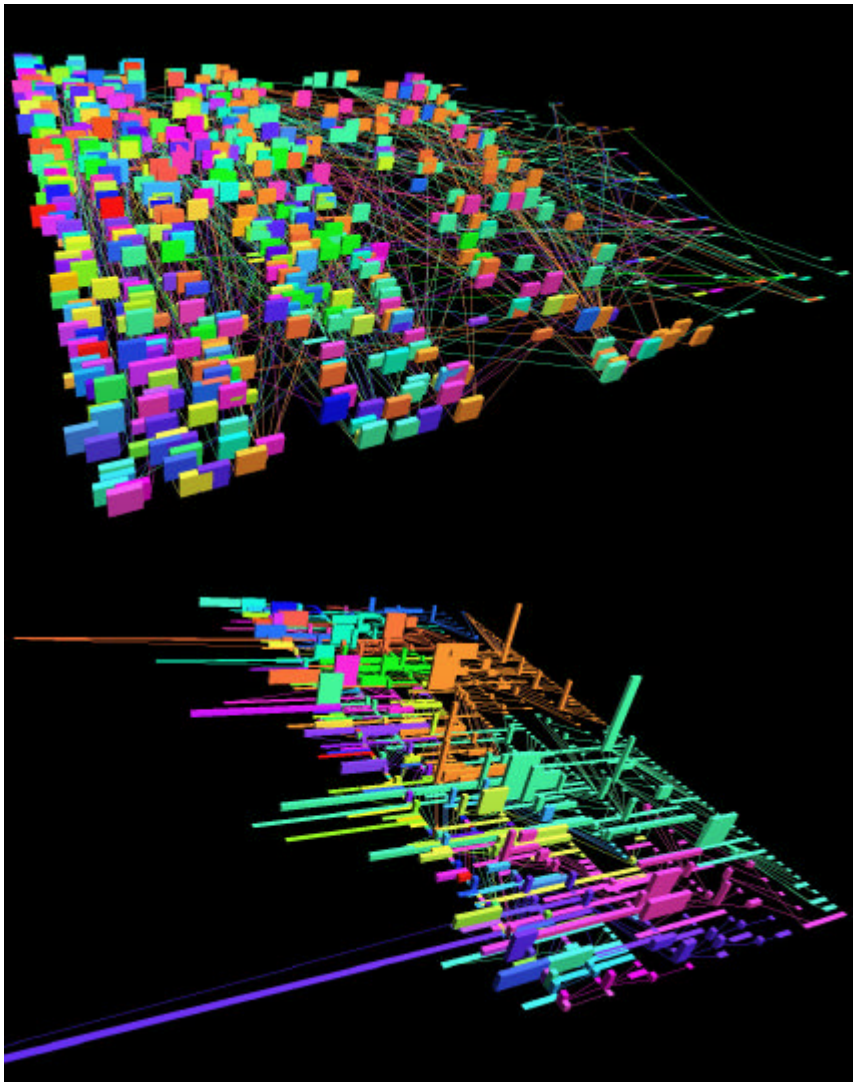

FIG. 5: Visualization of authoring and time information: Two Phase(x) outworld views showing the time of submission (top) and the duration and number of dependents (bottom) for each work. Urs Hirschberg.

3.2.2 Links and organization

Relationships assist the user in organizing and browsing the information space. These can be defined explicitly in the information structure or recognized using data mining techniques. Such virtual relationships can connect information entities that are otherwise not obviously related. This connectivity enables users to broaden their views beyond the immediate context of recent activities in time and scope.

Explicit relationships serve to link entities according to semantic information. Fig. 6 shows a (Java) visualization of project partners and their organization into groups or companies. The interface allows the user to drag individual components in order to rearrange the layout, and provides visual mechanisms for specifying new or altering existing relationships. Furthermore, the interface can assist the user when arranging the layout by keeping related components together and avoiding components to overlap. A constraint-based implementation will enable the optimization of such a layout.

Alternatively, explicit and virtual relations can serve as attractors for the positioning of information entities. Further control can be provided through selective visualization and a characterization of the entities and relationships using weights. Size, color, intensity, or transparency can all be used to express a component's weight; a terminological control may serve the user to specify these weights (Engeli, 1998).

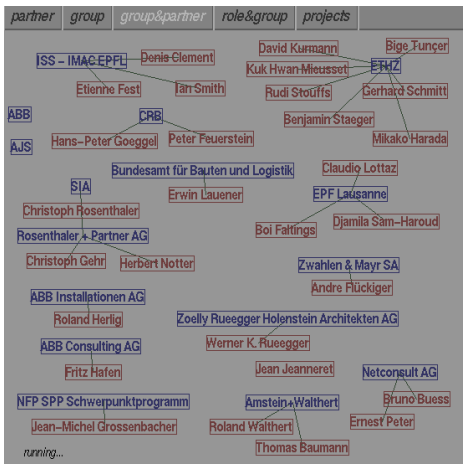

FIG. 6: Groups and partners visualization plug-in. Mikako Harada.

Fig. 7 illustrates a dynamic visualization of an entities hierarchy (in JavaScript). Entity titles are positioned with respect to a number of attractors, reflecting on both explicit and virtual relations. Different types of attractors are illustrated. The specification of a single category ('pipes') or a set of categories serves to attract entities that share one or more of the specified categories. Alternatively, two categories can be placed in opposition within a single attractor ('security <-> existing'); entities are pulled to either end of the attractor corresponding to their specified categories. Fig. 7 also shows an attractor based on explicit grouping relations ('high hierarchy low'). The interface allows the user to define, select, position, and resize different attractors into a single field. The entity titles oscillate about their equilibrium positions, so as to facilitate the recognition of individual entities within local clusters.

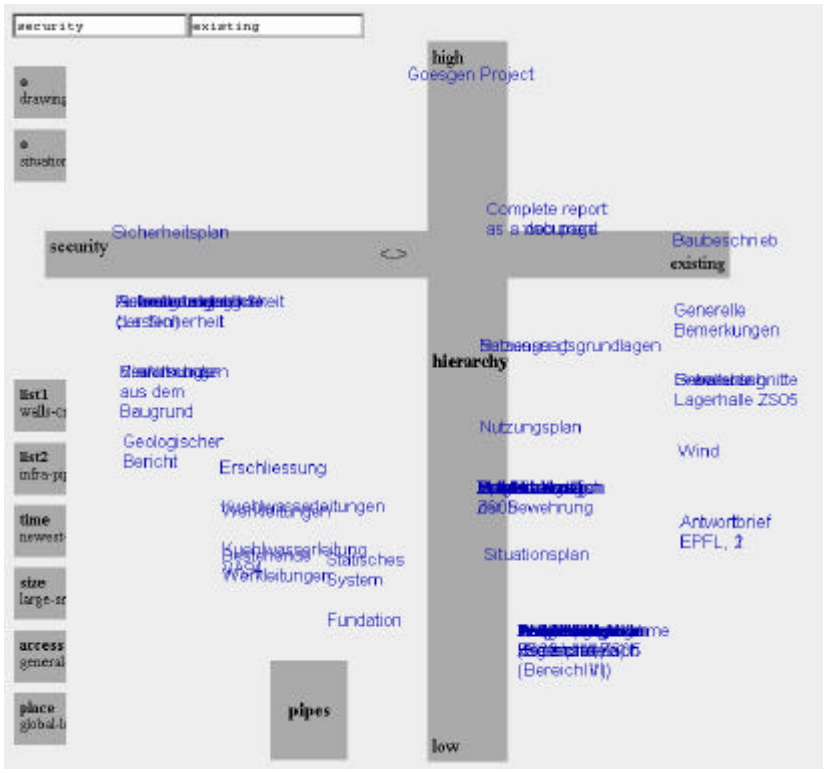

FIG. 7: Snapshot of a dynamic visualization of an entities hierarchy using attractors. David Kurmann.

3.3 Prototype framework and implementation

The environment's multi-tier architecture (Fig. 8) includes a service-based application server, a JDBC bridge to the database, an object-oriented middleware (implemented in Java), and dynamically linked software components to extend the environment's functionality. On the client-side, the prototype interface is developed in HTML and JavaScript, supporting an easy adaptation of the interface to particular needs or preferences.

The database does not serve as a central repository for documents created by the partners, instead it supports an information management system with the purpose of making project information accessible to all partners. Documents can either be referenced as URLs or uploaded to an HTTP-accessible directory. Security is provided through passwords for user authentication and digital signatures for the authentication of individual software components.

In order to alleviate the bottleneck of the Web, the basic configuration can be extended with an additional tier in the form of a *webtop* server (Gupta et al, 1998) that supports data caching in memory and duplicates most of the services on the application server, except for those that require database access. Ideally, such a webtop server can be installed at every partner company. Push technology, in the form of events and event handlers, ensures that all environment components are informed of changes in the project database.

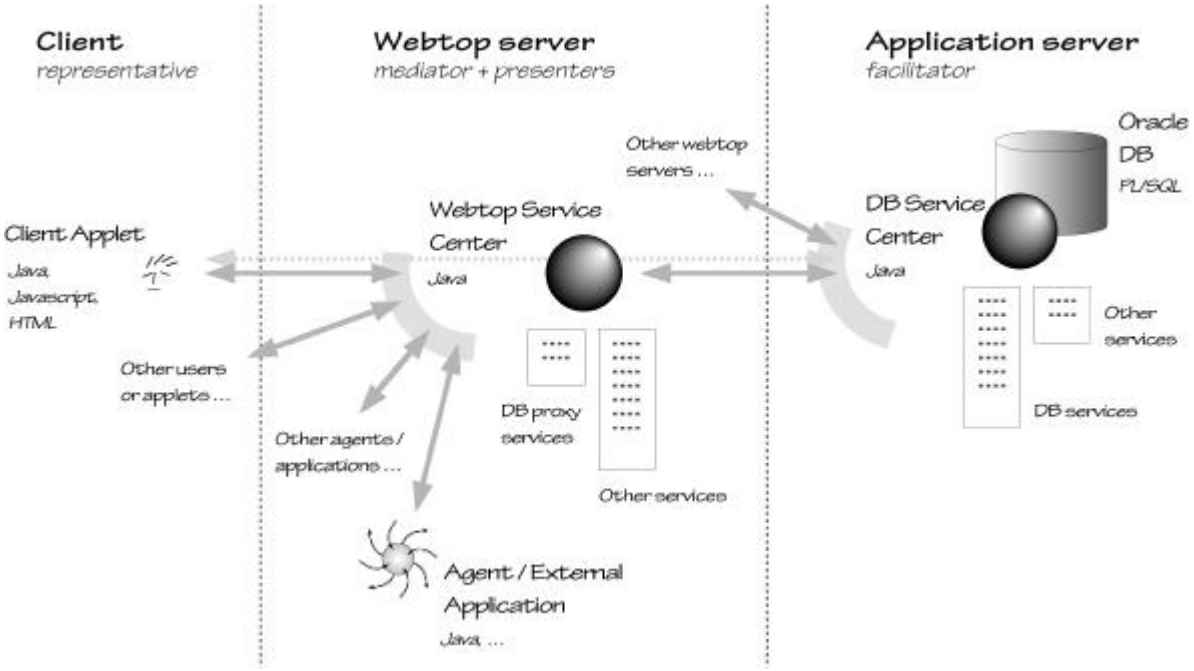

FIG. 8: Overview of the ICC architecture.

The prototype interface is accessible from http://iccs.arch.ethz.ch/.

3.4 Discussion

The ICC environment is being tested and evaluated both in-house and independently by the project's industry partners. The research team uses the environment for information management and publishing within the context of this research project, and evaluates the environment on realistic examples supplied by an industry partner. Independent evaluation is provided for in an educational and design collaboration involving different universities, and is considered by an IT partner in its use as a multiproject information environment and by a building engineering partner to support the management of a forthcoming construction project.

While the project's industry partners are very enthusiastic about the ICC environment and its adoption into practice, the fragmented and conservative nature of the AEC industry (in Switzerland) forms a serious obstacle to any practical use. The very essence of the environment, its support for collaboration, proves to be a hindrance to its adoption at the same time. Although the environment has been designed from the very onset keeping in

mind that not all collaborative partners may choose to participate, the environment's success still hinges on a concerted adoption by a number of partners. Furthermore, time constraints and delays that are all too common in construction projects obstruct the learning and adoption of new technology.

Nevertheless, the adoption of this environment, even if used only for data sharing and information gathering, yields clear advantages in time and information access. Time constraints and a lack of information can lead to errors and higher costs: a time constraint may impose the start of an activity before the necessary approval or other related information has arrived (see also section 4.1). Instead, electronic data sharing gives partners instantaneous access to published information.

Data sharing and exchange is only a narrow aspect of collaboration and such an environment should offer additional incentives to ensure a prominent place in the collaborator's tool set. These incentives can be presented in the form of visualizations that provide the partners with a better understanding and support during collaboration. Such context-sensitive and process-specific functionality is best captured in plug-ins and extensions to the environment that can be adopted and activated by the user when appropriate.

4. COLLABORATIVE DESIGN USING CONSTRAINT SATISFACTION TECHNIQUES

One extension to the ICC environment implements a constraint-based approach to collaborative design. It is possible to augment traditional design that uses point solutions by specifying requirements through constraints. Rather than considering design parameters and constraints exclusively as abstract data entities, taking semantics into account through constraints enables advanced support for negotiation and decision making in collaborative design.

When data transmission is augmented with explicit project constraints, collaboration systems can provide much additional engineering support. More specifically, representation, exchange, and manipulation of constraints have the following advantages:

- A description of solution spaces is available
- Artificial conflicts are avoided
- The initiator of a change retains responsibility for maintaining consistency
- When changes in requirements occur, much work can be reused
- Negotiation is guided within feasible spaces

More details and an illustrative example are given below.

4.1 The traditional point design approach

In traditional approaches to collaborative design, responsibilities for subtasks are distributed among several project partners. Collaborators determine a single solution to this subtask and meet again in order to negotiate for the integration of all partial solutions. This negotiation can become very difficult as conflicts are likely to arise due to premature decisions collaborators had to take while determining the single solutions.

For example, Table 1 illustrates the evolution of the values for four variables as project partners apply constraints related to their goals for the task. It refers to geometrical parameters of a beam with holes for passing ventilation ducts. Parameter *d* is the hole diameter, *e* is the hole spacing (center to center), *h* is the height of the beam, and *x* is the distance to the first hole as illustrated in Fig. 9. Each row in Table 1 represents one iteration; the partner named in the row initiates the change and changed values are typeset in bold. This example was inspired from the design, fabrication, and erection of a steel-framed building in Geneva, Switzerland. More details are provided in section 5.

TABLE 1: Possible negotiation process for the dimensioning of the beams in Fig. 9. Numbers are in mm.

	d	*e*	*h*	*x*
Architect	550	650	650	500
Steel fabricator	550	**900**	650	**1100**
Engineer	**200**	900	650	1100
Architect	200	900	650	**1000**

Ventilation subcontractor	**450**	**800**	650	**600**
Engineer	**400**	**900**	**730**	**700**
Steel fabricator	400	900	730	**800**
⋮	⋮	⋮	⋮	⋮

From Table 1 it is not clear if these iterations will ever converge on values that are acceptable to all partners. In reality, the steel contractor had to assume values before these iterations had terminated in order to satisfy the construction schedule. Several thousand dollars were wasted because eventually another partner refused the assumed values.

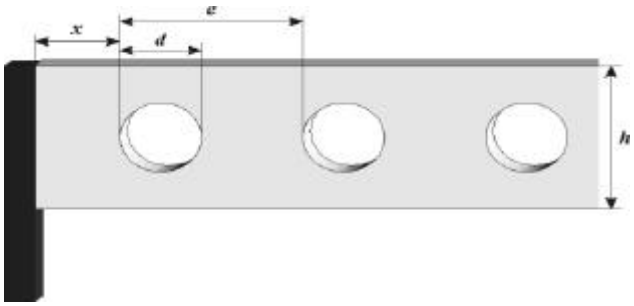

FIG. 9: Beam with holes for ventilation ducts.

4.2 Augmenting single solutions by solution spaces

Constraint Satisfaction Problems (CSPs) offer the possibility to calculate, represent, and manipulate solution spaces. Through departing from traditional point solutions (one value for each variable), CSPs augment the amount of information available for subsequent decisions. For example, CADRE (Hua et al, 1996) and IDIOM (Smith et al, 1996) use constraint solving on constraints on geometric parameters to enhance apartment floor layout plans, thereby facilitating adaptation of cases taken from previous designs.

Use of CSPs helps delay decisions for variable values until these become essential for the completion of the project. When premature decisions are reduced, information related to possible alternatives is retained. This is a variant of the least commitment paradigm often employed for planning tasks. In the automotive industry, decision delay strategies have already been adopted by major manufacturers (Ward et al, 1995), while it is a rather new concept in the AEC industry (Lottaz et al, 1999).

The specification of project requirements using mathematical expressions, i.e., equalities and inequalities, makes information explicit that may be invisible at first glance. Not only can conflicts be detected automatically but causal links can be deduced from the structure of the CSP. Moreover, solution spaces help to understand crucial tradeoffs and therefore support informed decision-making.

4.2.1 Conflict detection using local consistency techniques

Many conflicts in traditional collaboration arise from forced early decisions related to the values of variables. Often, there is no real conflict even though a negotiated change between two partners does not converge to an acceptable solution for all partners. When negotiating over single values for parameters, collaborators provoke artificial conflicts that lead to needless iterations of negotiation without revealing situations where actually no solution can be found (see section 4.1).

When collaborators use constraints to express their requirements, such artificial conflicts do not arise because no early decisions are taken. Moreover, the elaboration of constraints for a collaborative design task can help to detect real conflicts at early stages. When important requirements are expressed as mathematical relations, real conflict due to diverging design goals can be detected, even though the formalized information may not yet be complete. In this case, the collaborators have to compromise on a higher level before the work can go into a reasonable direction, revising certain goals to be reached with the design at hand.

For the ventilation example, a conflict may arise when the architect imposes the use of only two ducts for ventilation for cost reasons, while the engineer limits the size of the holes for structural integrity, and the ventilation subcontractor requires large ventilation capacity. Local consistency techniques may detect such a

situation very quickly although detecting the problem by hand may be difficult. When attempting to trace down the cause of the conflict, the structure of the CSP can show hidden dependencies and exploring these may lead to the discovery of conflicting goals.

4.2.2 Computational support for negotiation

When designers provide constraints that specify the requirements for their subtask, constraint satisfaction techniques can approximate the space of alternative solutions. In collaboration such solution spaces have the potential to simplify negotiation. When a solution space is known for each subtask, any solution for the whole task must be in the intersection of the solution spaces of the subtasks. This helps to choose alternative solutions for subtasks in a consistent way.

As such, negotiation is not avoided, but knowledge of the shape of the solution spaces encourages partners to be less restrictive during negotiation because any choice of parameters within the solution space guarantees that the specified requirements remain verified. Therefore, solution spaces provide useful support for negotiation.

Simplifying the problem concerning the dimensioning of the holes for ventilation ducts, the collaborators could specify constraints as shown in Table 2. Fig. 10 shows a three-dimensional projection of the corresponding solution space. As soon as this solution space is computed the collaborators can deduce from its shape where to search for solutions.

TABLE 2: Simplified constraints for the dimensioning of holes for ventilation ducts. Numbers are in mm.

Collaborator	*Constraints*
Architect	x < 1000, 600 < e < 1200
Engineer	x > 2d, d < 400, e > 900
Ventilation subcontractor	d > 300
Steel fabricator	x > 700, e > d + 50

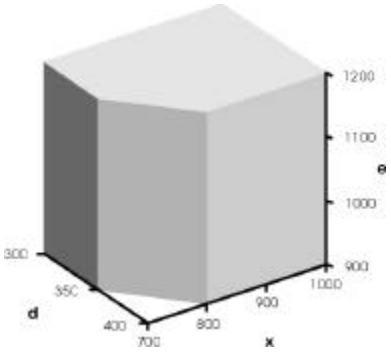

FIG. 10: A three-dimensional projection of the solution space for the dimensioning of beams with holes for ventilation ducts. Numbers are in mm.

Projections of the solution space as shown in Fig. 10 are also appropriate in more complex cases to visualize tradeoffs in order to understand, where compromises have to be made. When several optimization criteria have to be considered, projecting the solution space on these criteria supports good decision making.

4.3 *Space*Solver – an Internet-based CSP-tool set

*Space*Solver is a constraint solver for continuous CSPs based on the Internet. It provides extensions for collaboration. The constraint satisfaction techniques applied are best described in (Sam-Haroud, 1995, Sam-Haroud and Faltings, 1996). It provides an interface to the Web and is therefore well adapted to implement the collaborative design approach presented here. Its use involves a solving process in four phases:

- Specify parameters and constraints
- Perform an algebraic reformulation into ternary constraints
- Convert symbolic constraints into a spatial representation

- Compute consistency

This section describes *Space*Solver's functionality as a stand-alone application, while several issues raised in section 3, such as authoring information, automatic generation of links between data entities and visualization are taken into account. *Space*Solver is publicly available at http://liawww.epfl.ch/~lottaz/SpaceSolver/.

4.3.1 System architecture

*Space*Solver is an Internet-based application. It relies on the Common Gateway Interface (CGI) and performs most of the computation on the server. Its system architecture is illustrated in Fig. 11. Any Web browser can be used on the client-side, while on the server-side a dedicated Web server handles data management tasks and permanently communicates with various *Space*Solver modules. These modules include a *Symbolic Manipulator* for rewriting CSPs, a *Constraint Converter* for the generation of the spatial representation of constraints, a *Consistency Solver* to compute several degrees of consistency and a *VRML Generator* to visualize constraints and consistent spaces.

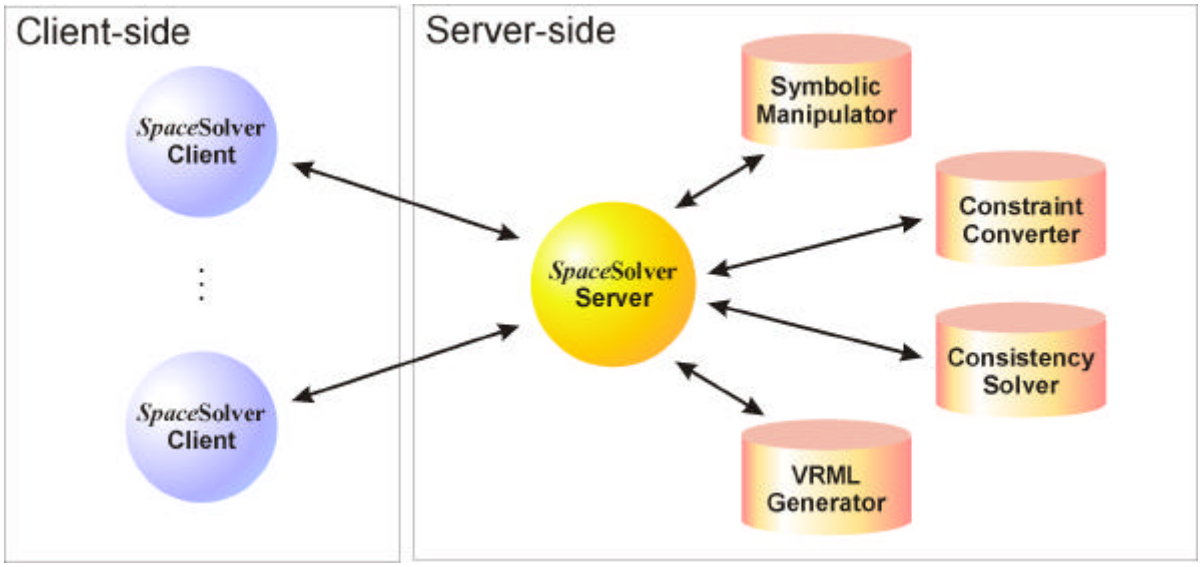

FIG. 11: Space*Solver's system architecture.*

The *Space*Solver server is implemented using the popular Apache Web server. CGI scripts compiled into the Apache server allow for efficient access to those scripts written in PERL and call the other modules using UNIX system calls. The *Symbolic Manipulator* is a collection of Maple V scripts. Maple V is a powerful symbolic algebra package. The communication between the server and the module is established using bi-directional pipes. The algorithms used for reformulation of the CSPs are described in (Lottaz, 1999). The *Constraint Converter* is implemented in C++. It reads the symbolic representation of a CSP in a file and generates octrees which represent the feasible spaces of the constraints (Samet, 1990). *Space*Solver's *Consistency Solver* implements the algorithms for arc-, path- and (3,2)-relational consistency as proposed by Sam-Haroud and Faltings in C++. Finally, the *VRML-Generator* reads files containing octree representations of constraints or solution space approximations and generated 3D-models in VRML.

4.3.2 Basic functionality

*Space*Solver provides support for a subset of the information aspects presented in section 3.1: information entities are restricted to variables and constraints; these are collected and organized by project; authorship is maintained; and relations between constraints and the variables these involve are established.

User registration and authentication: *Space*Solver is free to use. However, user registration and authentication is needed in order to distinguish the data from different users and manage access rights to projects. *Space*Solver maintains the data of registered users, allows them to create projects, and lets them participate in active collaborative projects.

Specifying constraints and variables: After login the user can choose a project from the selection list at the top of *Space*Solver's entry screen, initiate a new project, or select the page to adapt *Space*Solver's layout (Fig. 12). The *Navigator Frame* in the blue area on the left-hand side and the *Work Frame* on the white background at the right-hand side both update according to the choice made. The *Navigator Frame* allows users to switch between the different phases in the solving of the constraint satisfaction problem: a link is provided for each phase for which data is available. The *Work Frame* presents information on the creator of the project, the project's name, its abbreviation, and the definition of its corresponding constraint satisfaction problem.

Users can write their constraints in the text area provided, using Maple's format for mathematical expressions. Blank lines and comments may be included: anything to the right of a #-sign is ignored. When submitting a set of constraints, *Space*Solver automatically generates a table for the variables, with one row for each variable it finds in the users' constraints. This table allows the user to specify the range of valid values and a default value, as well as a small description for each variable.

Algebraic reformulation and submitting projects: Upon completion, the user needs to submit the specifications to the *Space*Solver server. The last choice to be made is the method for the algebraic reformulation adopted for the subsequent algorithms. The most important task of reformulation is to express the CSP exclusively in terms of ternary constraints. A ternary constraint is an equality or inequality, which involves at most three variables. This is done by introducing auxiliary variables for intermediary results in expressions.

Before adding auxiliary variables *Space*Solver can be asked to remove intermediary variables in order to keep the number of variables in the reformulated CSP low. The methods provided eliminate constants or intermediary variables, which depend on up to three variables. Variables are only eliminated if the CSP rewritten in ternary form can be expected to contain fewer variables through the elimination.

After the reformulation, a summary of the project is presented in the *Work Frame*. This summary includes the reformulated constraints from all collaborators, the remaining variables, the removed intermediary variables, and the added auxiliary variables.

4.3.3 Visualization

Section 3.2 recognizes visualization as an important means to improve the understanding of project information. *Space*Solver applies this to constraint information by allowing interactive analyses of the shape of constraints and potentially feasible spaces.

Generating spatial representations of total constraints: Whereas constraints are specified in a symbolic form as mathematical expressions, the consistency algorithms implemented in *Space*Solver use explicit spatial approximations of the feasible spaces according to these constraints. This conversion is done automatically and the user can choose its precision. Constraints involving the same set of variables are collected into one single constraint called a total constraint. A total constraint is the conjunction of all constraints involving the same set of variables.

When the generation of the spatial approximation of the feasible spaces for the total constraints is completed, the user is given the opportunity to analyze the characteristics of the constraints in a visual manner. VRML models of the feasible regions of the total constraints are generated and the collaborators can use a VRML plug-in to their Web browser, or any other VRML viewer, to analyze its spatial structure (see Fig. 13).

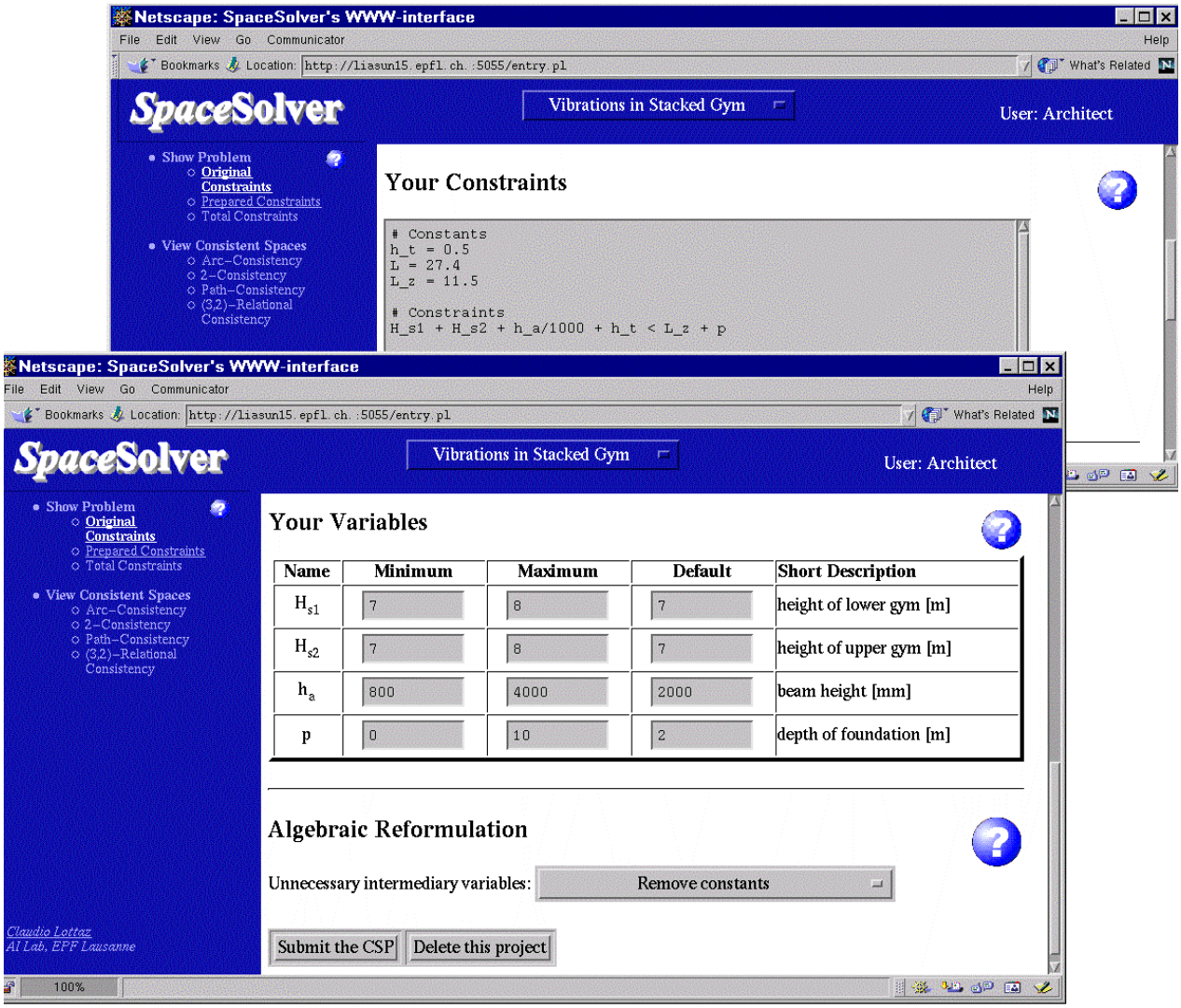

FIG. 12: Constraints are entered in a common format, used in the algebraic computation package Maple V.

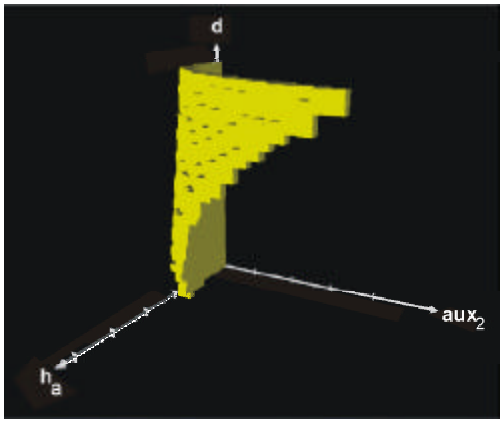

FIG. 13: Approximation of the total constraint $aux_2 = 3.37\ h_a\ d$ and $h_a <= 100\ d$.

Computing consistency: Several consistency algorithms for different degrees of consistency are provided. Weak consistency algorithms only check for very local inconsistencies while stronger consistency algorithms

can provide global consistency in special cases. *Space*Solver provides facilities to compare the different results in an intuitive fashion. Fig. 14 shows the same projection of the approximation of a solution space computed by different consistency algorithms.

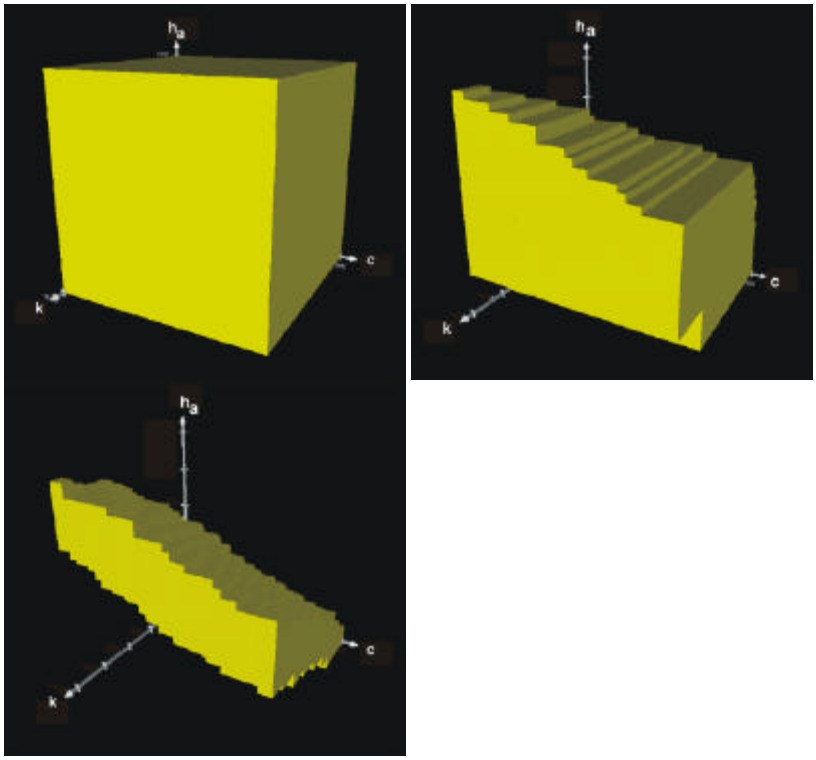

FIG. 14: Three-dimensional projection of solution space approximated by arc-, path-, and (3,2)-relational consistency (from left to right).

After the calculation of a certain consistency algorithm has finished, any one-, two-, and three-dimensional projections of the corresponding consistent space can be visualized. Similar to visualizing total constraints, VRML models are generated and displayed using an adequate viewer. Moreover, for each variable *Space*Solver can provide a set of intervals that represents potentially feasible ranges for this variable.

Such visualization allows the analysis of dependencies, which were not explicit in the original formulation of the problem. It is thus possible to visualize tradeoffs inherent in a problem but not stated explicitly.

4.3.4 Collaboration extensions

Collaborating simultaneously on one project: Several collaborators are allowed to participate in the same project. All collaborators maintain their own file of constraints, but they can share variables, i.e., a variable can be implied in constraints by different collaborators. The creator of a project specifies who is allowed to contribute to the project (Fig. 15). Collaborators can be added and removed at any time. Moreover, the creator of a project can also change the project's name and abbreviation.

Finding information about collaborators' constraints: The constraints and variables submitted by other collaborators can be investigated through links generated by *Space*Solver. For every collaborator a link is provided that brings up a page displaying the constraints posted by that collaborator.

Since certain variables will be shared, collaborators must be able to find which variables are already defined and use these common variables. *Space*Solver provides a summary of all variables defined with their minimum, maximum, and default value, as well as a short description.

Collecting data into one constraint satisfaction problem: The constraints and variables defined in such a distributed way are collected into a constraint satisfaction problem whenever collaborators submit their data. The specification of project requirements is thereby not necessarily complete. Nevertheless, *Space*Solver collects all

data so far submitted to the current project, shows it on the *Prepared Constraints* page, and allows for further possibly preliminary analysis.

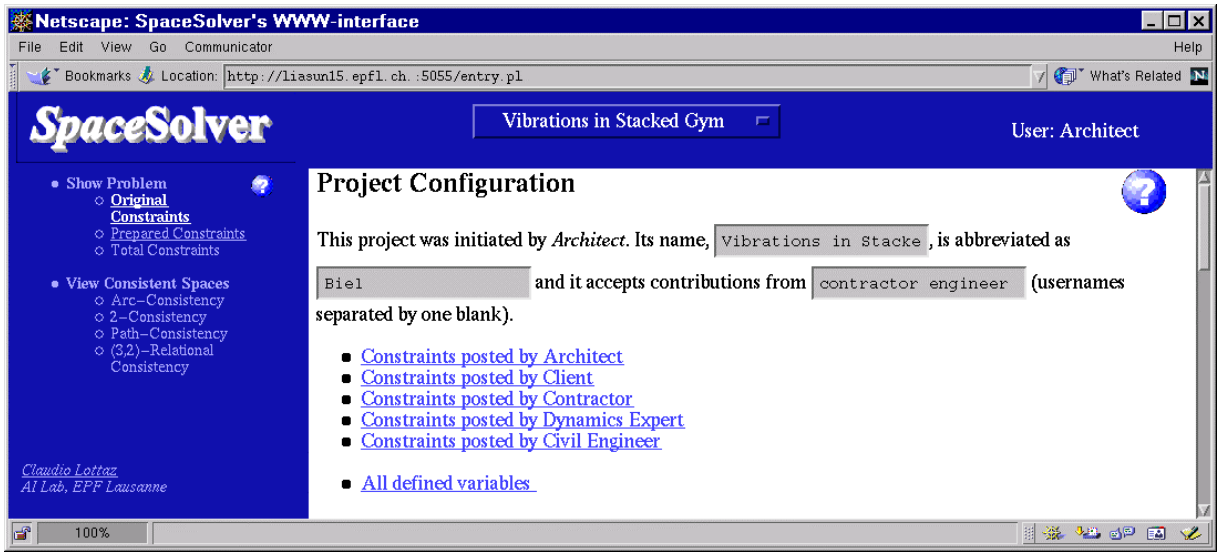

FIG. 15: Space*Solver allows collaborators to work simultaneously on a project.*

4.4 Discussion

We suggest constraint-based support for collaborative design. The use of solution spaces makes collaboration more efficient through avoiding artificial conflicts and providing computational support during negotiation.

*Space*Solver is a tool set for analyzing and solving constraint satisfaction problems on continuous domains. It provides an intuitive user interface including the specification of constraints in usual mathematical writing and an interactive three-dimensional visualization of constraints and solution space approximations. Moreover, *Space*Solver provides the necessary extensions for collaborative work and thus implements the solution space approach to collaboration, it is in fact a communication platform for highly structured data.

The current implementation is restricted to numeric CSPs expressed using mathematical expressions. The treatment of discrete variables and constraints would need some more development of appropriate user interfaces as well as adaptations of the solver. Also disjunctive problems cannot be treated in the current version, integration and further development of recent research into dynamic constraint satisfaction (Gelle, 1998) would be necessary.

Although some work in collaborative design and concurrent engineering proposes constraints to capture requirements, there is only little research into the explicit use of solution spaces. Most collaboration approaches using constraints concentrate on constraint checking e.g. (Klein, 1997) or use local constraint propagation to suggest potential solutions when conflicts occur e.g. (Bahler et al, 1995). Spaces of solutions have been used by Darr and Birmingham (1996). In their work, local consistency techniques help to prevent combinatorial explosion in configuration design problems. However, all of these approaches aim to determine single optimized solutions for collaboration projects, while our approach proposes to support cooperative negotiation using exploration and visualization of solution space approximations.

*Space*Solver provides a way of communication with strict semantics. Collaborators must provide information in formal languages, i.e., mathematical expressions which enable rapid syntheses to be carried out. However, it was observed that the restricted form of communication imposed by *Space*Solver is not sufficient for efficient collaboration. During collaborative work the need for communication with free-form documents arises very quickly. For instance, short descriptions are not sufficient to define a variable accurately. Drawings and textual documents may also be needed to explain and justify constraints. Therefore additional information on a platform such as the ICC environment is needed to facilitate explanations and discussions.

5. INTEGRATING SPACESOLVER AND THE ICC ENVIRONMENT

We observe that neither a completely free communication platform such as the ICC environment (section 3) nor the restricted way of communicating as suggested by *Space*Solver (section 4) provides optimal support for collaborative design. However, both improve the communication between collaborators on complementary aspects. Therefore, the combination of the two is of interest.

5.1 Working together

An extension to the ICC environment enables *Space*Solver to automatically store and update a constraint satisfaction problem in the project database of the ICC environment (Fig. 16). The resulting information space contains an entity for each constraint and each variable in the CSP. Representing the bipartite graph of the CSP, each constraint entity relates to all variables it is defined over while each variable entity has relationships to all relevant constraints it is involved in.

This extension consists of a small Java applet that provides the necessary functions to create and update constraint and variable entities in the project database through JavaScript. It also allows *Space*Solver to navigate the information space remotely by specifying the entities to be displayed in the user's ICC interface. Links in the *Space*Solver interface update the ICC interface accordingly. Alternatively, the user can explore the CSP in the ICC interface by following the relationships between the constraint and variable entities.

Finally, CSPs are enriched using features of the ICC environment. New and existing documents or entities can be linked to provide definitions and explanations to constraint and variable entities created by *Space*Solver. These entities themselves can be modified with new attribute information and, as long as such changes do not overwrite information provided by *Space*Solver, the ICC environment will maintain them even when *Space*Solver updates the constraint and variable entities.

5.2 Evaluation examples

Two building engineering projects have been selected in order to illustrate the benefits of combining both environments for collaborative design.

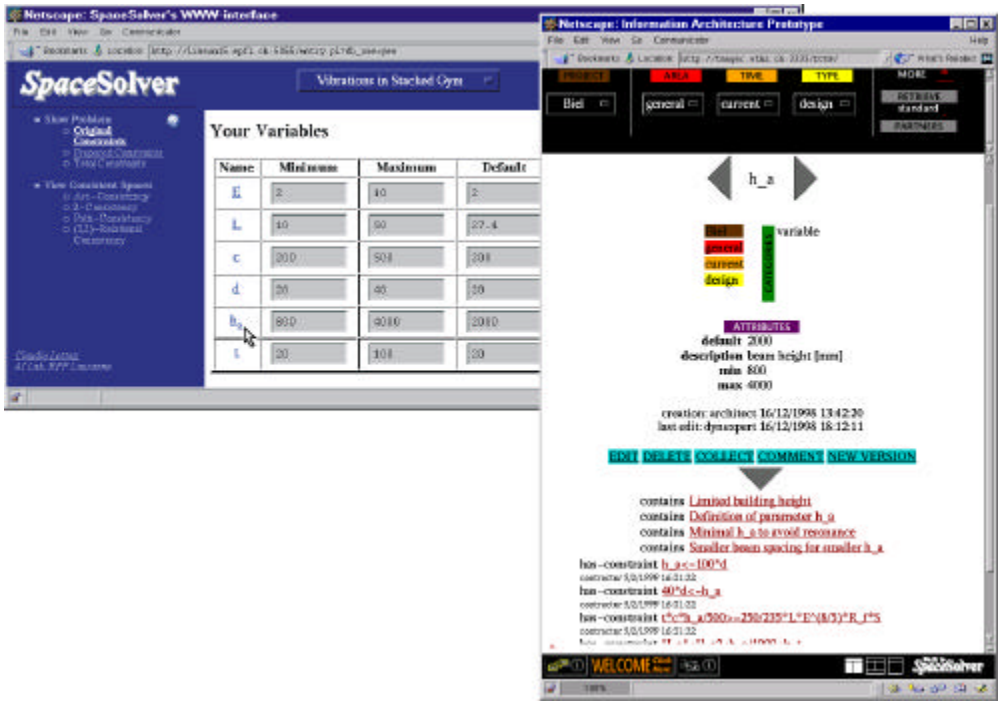

*FIG. 16: Space*Solver *(top) and ICC (bottom) view of a simple constraint satisfaction problem.*

5.2.1 A steel-framed computer building

The project: This example is inspired by an existing building in Geneva, Switzerland, where poor collaboration resulted in higher than necessary construction costs. In this and other buildings that house various types of servers and other large computers, ventilation requirements are important design criteria. Good ventilation maintains satisfactory operating temperatures and this leads to greater equipment reliability. Therefore, space has to be reserved for ventilation ducts. However, increasing the building height has a strong impact on construction and operating costs.

Adopting beams with holes allows the passage of ventilation ducts where beam material is not used efficiently, thereby providing effective solutions at reasonable costs (Fig. 17). The choice of the number of holes and ducts, the spacing of the beams, and the beam height, hole diameter, and other geometric parameters, is however not easily established. This issue generated much discussion and negotiation between the collaborators on this building project.

The project collaborators that are most concerned with this issue are the architect, the engineer, the steel fabricator and the ventilation subcontractors. Architects generally aim for an aesthetically pleasing distribution of holes and good proportions of hole size with respect to other dimensions. Engineers typically require few small diameter holes so that beam strength is not compromised. The steel fabricator prefers high values for hole spacing and no hole proximity to connections in order to avoid effects of stress concentrations caused by the holes. Finally, ventilation subcontractors want large, closely spaced holes everywhere so that they can accommodate later changes most easily. Such conflicting goals are common in every construction project.

FIG. 17: Construction site of the steel-framed building example. Holes in beams hold ventilation ducts.

Expressing requirements with constraints: When collaborators want to benefit from solution spaces as suggested in section 4 they must express the requirements they impose as constraints. The parameters involved in the equalities and inequalities must be defined precisely and the shared parameters in constraints of several collaborators need agreement upon their definition. For instance, variables involved in the project described are *x, d, e* and *h*. In order to define these without ambiguity, collaborators most likely refer to drawings like Fig. 9. Other non-geometric parameters such as c_V, the coefficient of air renewal, need some textual definition such as "which part of the air of the whole room is exchanged in one second."

While the project partners are working on the specification of their respective requirements, these definitions must be available and up to date. *Space*Solver only provides for a short textual description for each variable. When combined with the ICC environment, each variable has a corresponding information entity in the project database of the ICC environment and *Space*Solver provides a link to display this entity in the ICC interface. Using the information management facilities provided by the ICC environment, collaborators can attach clarifying documents to the variable entity, thus simplifying the specification of the requirements using constraints.

5.2.2 Vibrations in Stacked Gym

The project: This example is taken from a completed project to build two triple gymnastic halls in the city center of Bienne, Switzerland (Fig. 18). Due to limited space, one gym was placed on top of the other. This led to special constraints related to vibrations of the floor beams of the upper gym. In addition, the owners had

special requirements related to the position of these beams since these are used to fix equipment that is used in the lower gym. Local building laws include a building height restriction for this zone and, in addition, the building is situated on a near-surface aquifer. If foundations were placed below the water table, pumping and waterproofing costs would have been unacceptable. Therefore, a solution for squeezing two gyms between the building height limit and the water table, while avoiding vibration problems and meeting owner requirements, had to be found.

FIG. 18: Two triple gymnastic halls, one placed on top of the other.

Exploring solution spaces: After most of the collaborators have stated their constraints, *Space*Solver can provide an approximation of the solution space. This solution space can be projected on any set of up to three variables. Fig. 19 shows a projection of the solution space on three of the geometric parameters, which determine the shape of the floor beams of the upper gym. As long as the solution space approximations determined by *Space*Solver are not empty, collaborators can negotiate within these spaces.

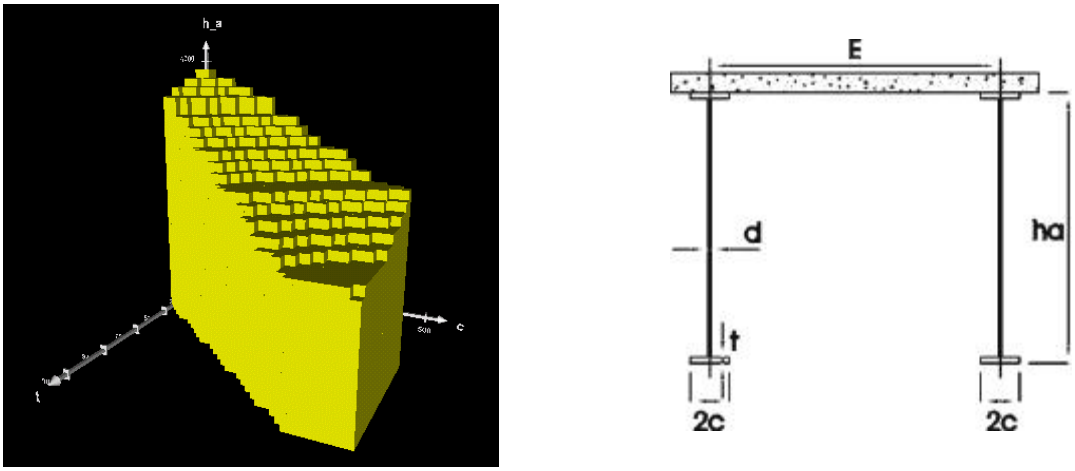

FIG. 19: Projection of solution space on half cover plate width (c) and thickness (t), as well as beam height (h_a). To the left: a cut through two beams.

Finding and resolving conflicts: A tradeoff between several design goals is often necessary. In such cases, *Space*Solver can determine whether a conflict exists where no choice for the parameters will satisfy all constraints. However, it is not able to determine the cause of the conflict; in fact, this task has been shown to be intractable in the general case. Nevertheless, the combination of *Space*Solver with the ICC environment provides some support for humans to search for the cause of a conflict and subsequently negotiate to resolve it.

Suppose that a conflict arises when the civil engineer introduces an additional constraint on *E* (the spacing of the floor beams). When *Space*Solver is launched in combination with the ICC environment, it provides a link for *E*

to display the corresponding entity in the ICC interface. This view includes a list of all constraints in which *E* is involved. Among these constraints the engineer finds *E > 2.5.* This constraint seems very arbitrary and might be the cause of the conflict. Therefore, the engineer follows the corresponding link and the resulting constraint entity view reveals two justifications for this constraint: the architect needs the space for hanging lamps and the client asks for it in order to mount sports equipment such as rings and basketball baskets.

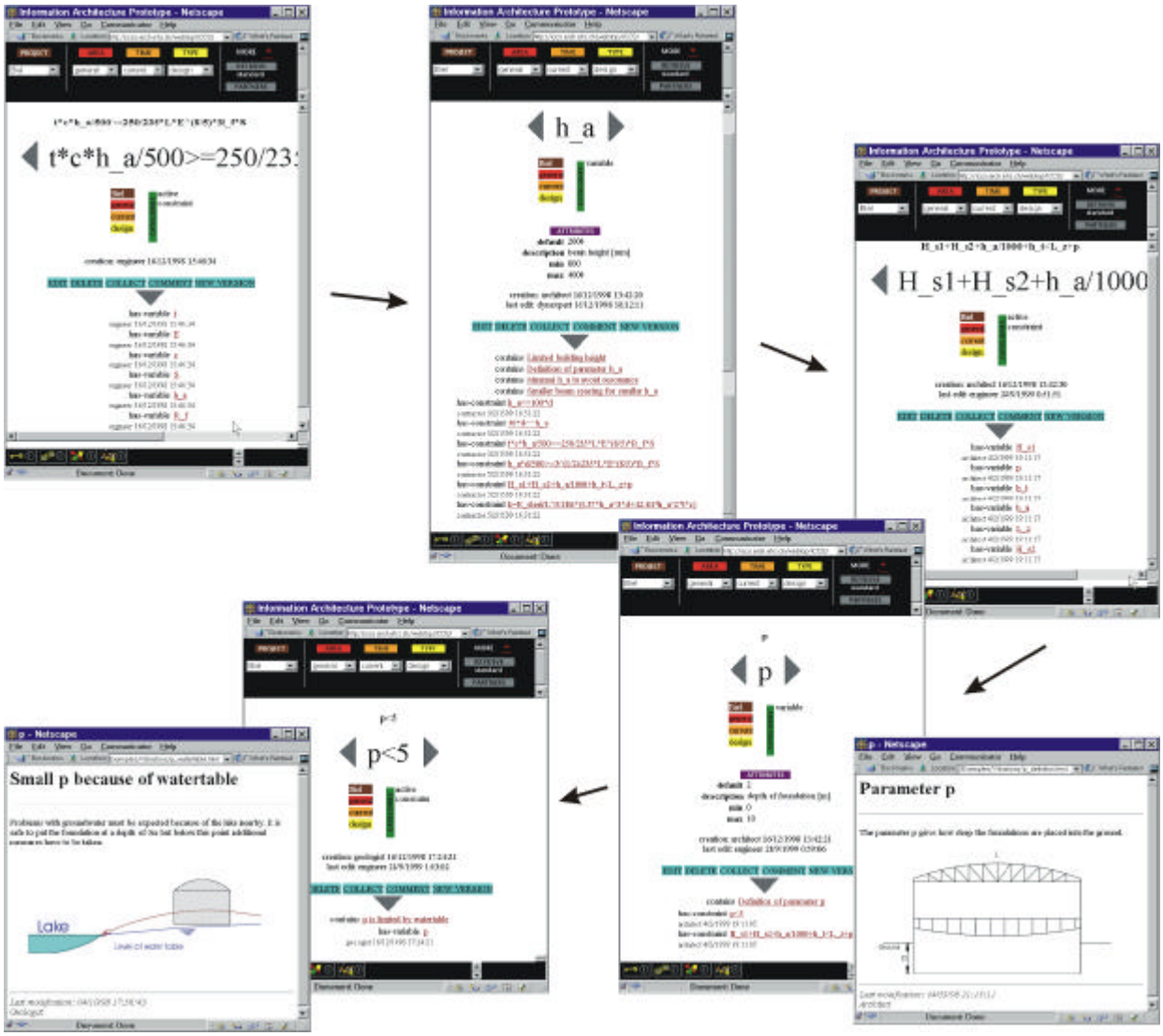

FIG. 20: Exploring the information space in order to find conflicts.

Since this constraint seems not to be negotiable, the engineer backs up to the *Space*Solver interface. He finds the link that allows the information about the constraint last added, the one which caused the conflict, to be displayed in the ICC interface. From the relationships to all variables involved in that constraint, the engineer follows the link to h_a (the beam height), because this parameter is most influenced from outside the engineer's own constraints. From the information for h_a, the engineer finds the constraint $H_{s1} + H_{s2} + h_a + h_t < L_z + p$, which seems critical for its high number of variables involved. The engineer then follows the relationship to p (the footing depth), because this parameter was not yet examined. The engineer finds the definition as well as the justification for its upper limit from the geologist. After this exploration of the information space (Fig. 20), including the screening of the definitions of the variables and the explanations of the constraints, and including the exploitation of the relationships between constraints and variables, the engineer starts to negotiate with the client about the height of the gyms (H_{s1} and H_{s2}). Finally the client accepts a slightly smaller height of the upper hall.

5.3 Integration architecture

When *Space*Solver is started for concurrent use with the ICC interface, both clients are started at once, each in its own browser window. A third client, without user interface and contained in the *Space*Solver browser window, incorporates the extension to the ICC environment and enables the synchronization of the other clients (and their servers) during the concurrent session. This extension searches for the interface client on the ICC network that is opened by the same user, and forwards all instructions from the *Space*Solver client running in the same browser to this interface. In this way, the two software packages can work together smoothly, even though the ICC application server and the *Space*Solver server are located in different geographical areas and do not communicate directly between each other.

5.4 Discussion

We suggest an integration of two complementary information and communication platforms. The technical integration allows the two servers involved to be located in geographically different places with no need for direct communication between them. The communication link is established locally between the *Space*Solver client and the ICC extension client operating within the same browser window, such that the other components, including the respective servers, behave exactly as in stand-alone use.

From a user's point of view the integration is especially smooth. The only interface changes are additional links in the *Space*Solver interface that allow the user to control the navigation and display in the ICC interface from the *Space*Solver client.

6. CONCLUSIONS

With growing complexity and shrinking construction time the need for efficient collaboration in the construction industry is becoming more and more important. Collaboration may be hindered by the lack of effective and efficient facilities for exchanging and organizing project information.

The ICC system implements an environment for the management and presentation of distributed, related data generated and exchanged in the context of a collaborative building project. It aims to augment the partners' current computing environment with support for information sharing and collaboration, in support of existing work processes and concurrent to existing applications. The ICC environment has been extended with tools to manage decision making and visualize the information structures that are built in a collaborative effort.

Collaborative design using constraint solving constitutes a complementary collaborative approach. Information with strict semantics, i.e., constraints and variables, supports decision making for collaboration. *Space*Solver implements this approach and provides support by making implicit relations between parameters visually explicit. Its consistency algorithms can detect conflicts and determine tradeoffs, assisting the user in making informed decisions. Moreover, *Space*Solver helps in adopting a least commitment approach. It has been shown that this approach has the potential to improve both the time for developing and building an artifact as well as its quality.

Integrating both environments provides additional benefits. Synergies have been validated in two projects from the construction industry. From a civil engineering point of view, working with constraints is a natural approach. The determination of solution space approximations allows simpler negotiation between collaborators and helps to detect conflicts. The ICC environment allows management of all information and documents generated and exchanged in the collaboration. These information entities may then be linked to constraints in order to explain important restrictions and decisions for an effective negotiation.

7. ACKNOWLEDGMENTS

This work, under the leadership of Gerhard Schmitt, is funded by the Swiss National Science Foundation, grant no. 5003-45357. For their role in the development of the ICC environment, the authors thank Bige Tunçer, Kuk Hwan Mieusset, Benjamin Stäger, David Kurmann, and Mikako Harada of ETH Zurich. Their specific contributions are as marked in the respective figure captions; the *Phase(X)* images are used by kind permission of Urs Hirschberg. In addition for providing invaluable help in modeling and analyzing the evaluation projects the authors would like to thank André Flückiger of Zwahlen & Mayr SA, and Denis Clément, Etienne Fest, and Yvan Robert-Nicoud of EPF Lausanne.

8. REFERENCES

Almeida L., Grilo A., Rabe L. and Duin H. (1998). Implementing EDI and STEP in the construction industry, *Product and Process Modelling in the Building Industry* (Amor R., editor), Building Research Establishment, Watford, England, 1-8.

Anumba C.J., Cutting-Decelle A.F., Baldwin A.N., Dufau J., Mommessin M and Bouchlaghem N.M. (1998). Integration of product and process models as a keystone of concurrent engineering in construction: the ProMICE project, *Product and Process Modelling in the Building Industry* (Amor R., editor), Building Research Establishment, Watford, England, 9-19.

Arb S.,von, Güntensperger T. and Schärer W. (1997). Integration von Aufgaben, Prozessen und Daten im Bauwesen - ZIP Bau, *Jahrbuch 1996*, Department of Architecture, Swiss Federal Institute of Technology Zurich.

Bahler D., Dupont C. and Bowen J. (1995). Mixed quantitative/qualitative method for evaluating compromise solutions to conflicts in collaborative design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 9, No. 4, 325-336.

Beetz K., Junge R. and Steinman R. (1998). The O.P.E.N. platform, *Product and Process Modelling in the Building Industry* (Amor R., editor), Building Research Establishment, Watford, England, 91-100.

Bowen J. and Bahler D. (1993). Constraint-based software for concurrent engineering, *Computer*, Vol. 26, No. 1, 66-68.

Buckley E., Zarli A., Reynolds C. and Richaud O. (1998). Business objects in construct IT, *Product and Process Modelling in the Building Industry* (Amor R., editor), Building Research Establishment, Watford, England, 117-130.

Cutkosky M.R. and Tennenbaum J.M. (1996). Collaborative engineering over the Internet, *Communications of the ACM*, Vol. 39, No. 9, 78-87.

Divita E.L., Kunz J.C. and Fischer M.A. (1998). Collaborative desktop engineering, *AI in Structural Engineering*, Lecture Notes in AI, Springer.

Engeli M. (1998). Information technology - Discovering new spaces for design, *FIDE'98 International Conference on First Year Architectural Design Education*, Faculty of Architecture, Istanbul Technical University, Istanbul, 3.31-3.42.

Fruchter R. (1996). Conceptual, collaborative building design through shared graphics, *IEEE Expert, Intelligent Systems & their Applications*, June, 33-41.

Gelle E.M. (1998). On the Generation of Locally Consistent Solution Spaces in Mixed Dynamic Constraint Problems, PhD-thesis No. 1826, Swiss Federal Institute of Technology Lausanne (EPFL).

Gupta A., Ferris C., Wilson Y. and Venkatasubramanian K. (1998). Implementing Java computing: Sun on architecture and applications deployment, *IEEE Internet Computing*, Vol. 2, No. 2, 60-64.

Herbert L. (1997). Collaborative design on global projects: Evolving needs in AEC industry. In preprints of *IFIP WG 5.2 Workshop on Formal Aspects of Collaborative CAD* (Maher M., Gero J. and Sudweeks F., editors), Key Centre of Design Computing, University of Sydney, Sydney, 11-14.

Hua K., Faltings B.V. and Smith I.F.C. (1996). CADRE: Case-based geometric design, *Journal of Artificial Intelligence in Engineering*, Vol. 10, 171-183.

Hirschberg U. and Wenz F. (1997). Phase(x): Memetic engineering for architecture, *Challenges of the Future* (Martens B., Linzer H. and Voigt A., editors) (Proceedings ECAADE conference, 17-20 September 1997, Vienna).

Khedro T. and Genesereth M. (1994). Progressive negotiation for resolving conflicts among distributed heterogeneous cooperating agents, *12th National Conference on Artificial Intelligence*, AAAI, 1994.

Klein M. (1997). An exception handling approach to enhancing consistency, completeness and correctness in collaborative requirements capture, *Journal of Concurrent Engineering Research and Applications*, 73-80.

Lottaz C., Clément D., Faltings B.V. and Smith I.F.C. (1999). Constraint-based support for collaboration in design and construction, *Journal of Computing in Civil Engineering*, Vol. 13, No. 1, 23-35.

Lottaz C. (1999). Rewriting numeric CSPs for consistency algorithms, *Workshop on Non-Binary Constraints at the International Joint Conference on Artificial Intelligence*, Stockholm, E:1-E:13.

Mokhtar A., Bedard C., and Fazio P. (1998). Information model for managing design changes in a collaborative environment, *Journal of Computing in Civil Engineering*, Vol. 12, No. 2, 82-92.

Ndumu D.T. and Tah J.M.H. (1998). Agents in computer-assisted collaborative design, *AI in Structural Engineering*, Lecture Notes in AI. Springer.

Peña-Mora F., Siriram D. and Logcher R. (1995). Design Rationale for Computer-supported Conflict Mitigation, *Journal of Computing in Civil Engineering*, Vol. 9, No. 1, 57-72.

Peña-Mora F. (1998). A collaborative negotiation methodology for large scale civil engineering and architectural projects, *AI in Structural Engineering*, Lecture Notes in AI, Springer.

Peña-Mora F. and Wang C.Y. (1998). Computer-supported collaborative negotiation methodology, *Journal of Computing in Civil Engineering*, Vol. 12, No. 2, 64-81.

Petrie C.J., Jeon H., and Cutkosky M.R. (1997). Combining constraint propagation and backtracking for distributed engineering, In *Constraints & Agents*, Workshop at AAAI'97, Menlo Park, CA, AAAI Press, 76-82.

Preiss K., Goldman S. and Nagel R. (1997). *Cooperate to Compete: Building Agile Business Relationships*, Van Nostrand Reinhold.

Rezgui Y., Cooper G. and Brandon P. (1998). Information management in a collaborative multiactor environment: the COMMIT approach, *Journal of Computing in Civil Engineering*, No. 12, 136-143.

Roddis W.M.K. (1998). Knowledge-based assistant in collaborative engineering, *AI in Structural Engineering*, Lecture Notes in AI, Springer.

Roe A. (1999). New digital products boost collaborative design, *CADENCE*, February, 26-29.

Sam-Haroud D. (1995). Constraint consistency techniques for continuous domains, PhD thesis No. 1423, Swiss Federal Institute of Technology in Lausanne, Switzerland.

Sam-Haroud D. and Faltings B. (1996). Consistency techniques for continuous constraints, *Constraints*, Vol. 1, No. 1&2, 85-118.

Samet H. (1990). Applications of Spatial Data Structures, Addison-Wesley, Reading, Mass.

Scherer R.J. (1998). A framework for the concurrent engineering environment, *Product and Process Modelling in the Building Industry* (Amor R., editor), Building Research Establishment, Watford, England, 449-458.

Schmitt G., Stouffs R., Kurmann D., Tunçer B., Mieusset K.H., Stäger B. and Harada M. (1999). A tool set for the virtual AEC company, *http://iccs.arch.ethz.ch/*, Chair for Architecture and CAAD, Swiss Federal Institute of Technology Zurich.

Smith I.F.C., Stalker R. and Lottaz C. (1996). Creating design objects from cases for interactive spatial composition, *Artificial Intelligence in Design*, Kluwer Academic, Dordrecht, The Netherlands, 97-116.

Stouffs R., Tunçer B. and Schmitt G. (1998). Supports for information and communication in a collaborative building project, *Artificial Intelligence in Design 98* (Gero J.S. and Sudweeks F., editors), Kluwer Academic, Dordrecht, The Netherlands, 601-617.

Ward A., Liker J.K., Cristiano J.L. and Sobek D.K. (1995). The second Toyota paradox: How delaying decisions can make better cars faster, *Sloan Management Review*, MIT, Spring 1995, 43-61.

Wix J. (1997). Information models and modelling: standards, needs, problems and solutions, *The International Journal of Construction Information Technology*, 27-38.