CIRCLE INTEGRATION

by

Martin Fischer and John Kunz

CIFE Working Paper
Number 20

April, 1993

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

CIRCLE INTEGRATION

Martin Fischer[1] and John Kunz[2]

"Designing that building was like going around in circles."
Carlo Séquin

ABSTRACT

This paper proposes "circle integration" as a simple, testable approach to structure the integration of AEC software applications. In circle integration, each application is linked to exactly one predecessor and one successor application. Thus, changes made to the design or construction plan of a facility will be propagated automatically to all other applications. A pass "around the circle" thereby defines and completes one design iteration for one specialist, making the definition of design versions explicit and simplifying version management. We expect that specialists will have private copies of all applications on the circle for a particular project. With modern computers, a single pass will be very fast (minutes), and specialists will be able to evaluate proposed alternatives quickly and independently from multiple perspectives. Specialists will continue to use normal social conventions of their organizations and take their recommendations to design meetings. In these meetings, specialists will propose changes, identify conflicts through "quick runs of the circle," discuss these conflicts, make decisions, and accept new design versions. In summary, circle integration provides clear, accurate, rapid, and maintainable support for the creation and management of design versions.

---

[1] Asst. Prof., Dept. of Civ. Engrg., Stanford Univ., Stanford, CA 94305-4020

[2] Res. Assoc., Dept. of Civ. Engrg., Stanford Univ., Stanford, CA 94305-4020

Overview

The process of developing a constructed facility traditionally involves a number of phases with various high-level activities in each phase. The phases include pre-construction, construction, operation and maintenance, and decommissioning. We assume that the pre-construction phase encompasses a broad set of characteristics of a facility, including client requirements, specifications of functional systems, abstract features such as structural systems and circulation systems, dimensions, given and calculated loads, computed stresses, bills of materials, construction plans, schedules, and cost estimates. Among the disciplines that conduct pre-construction activities are architects, structural engineers, fabricators, etc. Integration of disciplines within a single phase is usually called "horizontal integration," whereas integration of two or more phases is usually called "vertical integration."

The principal purpose of horizontal integration is to make the entire pre-construction phase faster, more accurate, more consistent, and of higher quality than is possible without integration. The purpose of integrating the software applications used in pre-construction activities is to facilitate horizontal integration. Thus, measures of the success of software integration include the speed and quality of the pre-construction phase achieved and the overall project success. Successful vertical integration should contribute to the speed and effectiveness of coordination across the life-cycle phases. All integration should contribute to improved quality across multiple projects. As a step toward complete life-cycle integration, this paper discusses software integration of the pre-construction phase of civil engineering.

We consider the limited but representative set of pre-construction activities shown in Table 1. Often, the pre-construction phase includes additional activities, and completely different activities characterize subsequent phases. We consider the pre-construction phase activities because demonstration of their integration can serve as a basis for designing the integration of subsequent phases.

Each discipline has the responsibility to set and calculate values for some variables (its "independent variables") and to consider other values (for its "dependent variables"). The values of dependent variables might be given by other disciplines, by the client, or by regulatory agencies. For example, architects set story heights; structural engineers and architects together set the structural material. Estimators set the construction method (an independent variable for estimators) based on, among other variables, the story heights and structural material (dependent variables for estimators). Note that for a given discipline the status of variables as independent or dependent may change because of changing needs during the project life cycle, the preferences of the discipline's professionals, and changing project objectives. Thus, any integration approach must provide a clear basis for changing the status of variables from independent to dependent or vice versa.

CASE EXAMPLES

Throughout this paper we will refer to two real-world case examples. The next two sections introduce these examples and demonstrate several current problems in the project delivery process. They also outline opportunities for integration.

Illustrative case example 1: a modest university building

The Stanford University School of Engineering has recently completed the planning of a two-story 10,000 square-foot building with two case-study-style classrooms on the ground floor and computer laboratories on the upper floor. The planning team included approximately 25 people, consisting of consultants and various representatives from the university, such as facility managers and facility users (one of the authors was a representative for the client). In the course of the twelve months of the pre-construction phase, the group met initially every week or two. Later in the design development process, it met once or twice per month. Each meeting consisted generally of short presentations by the architect or other consultants and of discussions by client representatives. In some cases, the participants resolved issues during the meeting; in other cases, the participants asked the consultants to develop a new alternative for the next meeting.

Seen in retrospect, this process was very slow, and it allowed the development and evaluation of only a few alternatives. Since many of the university representatives had only a limited background in design and construction of the type of building planned, it took several months to define what the clients required, and many of the requirements changed up to the end of design development because the client representatives often disagreed on the exact specifications of important parameters.

The size of the classroom chalkboard, for example, was a continual discussion item. The planned seating forms a "U" around the diagonal of the roughly square classroom. The chalkboard will be mounted on a small wall that sits at the open end of the U and that cuts off a corner of the room. The question was how big to make the chalkboard and its support wall. One point of consideration was that if the chalkboard grew beyond a certain width, it would encroach on a wide door that allows direct access to an attractive grove of old oak trees. When this question was discussed, members of the group were unsure of how the size of the wall affects the (visual) "feel" of the room, how important a wide chalkboard is, what the value of the wide door is, and what size chalkboards are available on the market. Furthermore, they were unsure of what the cost impact would be if the two classrooms were not identical (e.g., if the chalkboard were moved forward and made longer in the room that did not face the oak grove). This example illustrates that design decisions do not only depend on what the designer envisions, but are often affected by the client's requirements and the availability of materials and systems. In particular, the example suggests the benefit of being able to evaluate the cost and value of alternatives quickly.

To help with the question of the visual feel of the room, one user representative built a 3D CAD model of the classrooms. The visual model significantly helped other representatives to understand layout options. However, as is now typical in industry practice, the CAD model itself was not directly linked to any other software applications. Thus, while visual what-if studies were performed through changes in the CAD model, they were not useful to help representatives to understand the construction, purchasing and operational tradeoffs concerning the different layout options.

In this paper, we propose an approach to structuring the integration of disciplines and activities to support basic engineering tasks. We do not address the objective of improving calculations of design parameter values, but we suggest that there is great potential value in a tool that calculates values and evaluates alternatives quickly from multiple perspectives. Such a tool will help designers to elucidate different client requirements and to enable project teams to converge on desirable alternatives much faster than today (Huber 1990).

Illustrative case example 2: a simple column

We use a generic cast-in-place reinforced concrete column of a type found on many commercial and industrial projects to illustrate ideas on integrating pre-construction computer applications. Figure 1 shows such a simple concrete column. The design allows on-site assembly of reinforcement bars and concrete pouring into a surrounding form.

We would like to illustrate the problem of inconsistencies between independent and dependent variables with the example of reinforced concrete columns for a warehouse project. As usual, the client set the budget (a dependent variable for the architect). The architect then selected the column material (cast-in-place reinforced concrete) and, given the client's requirements, set the column height to an atypically high value. Given these dependent variables, the structural engineer then selected a structural system and column dimensions and computed the member forces. Since these forces were incompatible with the local code requirements, the structural engineer added steel profiles and then sent the design to the estimator. This change increased the cost of the structural system (an independent variable for the estimator). Thus, a dependent variable for the architect (budget) and an independent variable for the estimator (cost) conflicted. Integration must provide mechanisms to identify such inconsistencies between dependent variables of one discipline and independent variables of others.

PRE-CONSTRUCTION ACTIVITIES

To illustrate integration requirements, this section elaborates on details of several pre-construction activities for case example 2 according to the summary of activities in Table 1. In each activity, professionals from a discipline solicit information from clients and recommendations from other disciplines (values for dependent variables). They then set and compute the values of the independent variables of their activity. After making the iterations necessary to satisfy local acceptability criteria, they represent their solutions in natural idioms common to the discipline and activity (e.g., drawings for architects, activity networks for schedulers) and make them available to other interested project participants.

- **Layout**: An architect receives requirements from the client regarding the purpose of a building. The architect then sets values for the location, material, and shape parameters of the building's columns. Often the architect and the structural engineer negotiate these decisions. The architect typically produces drawings to show the chosen solution. In later design iterations, the architect considers these drawings and the values set by other disciplines and their change requests.

- **Preliminary design**: Structural engineers receive the drawings and specifications regarding the use and location of columns produced by the architects. In addition to other tasks, they then assign and calculate the loads on the columns and set preliminary dimensions. Finally, they produce an initial structural drawing.

- **Analysis**: Structural engineers take a preliminary design solution and prepare the input to an analysis program, such as a finite element program. The analysis program calculates member forces that allow the structural engineers to evaluate the preliminary solution. If the member stresses and deflections seem acceptable, the engineers pass on the design responsibility to the next activity; if not they adjust the preliminary solution and re-analyze it.

The rest of the activities shown in Table 1 involve similar processing details. Some require additional information, such as resource availability and productivity rates for scheduling. The results of the pre-construction phase include drawings and a written specification of the project details, cost estimates and schedules.

Each activity normally includes the steps listed below. We give accompanying examples for a structural engineer, but each discipline has similar detailed steps.

- **Get inputs**, including functional requirements (e.g., use of space) and acceptability criteria (e.g., cost) from the client, acceptability criteria (e.g., tolerable stresses) from the discipline itself, and project data from other pre-construction activities (e.g., floor and window layout set by the architect).

- **Set and compute values of independent variables** (e.g., set concrete strength).

- **Evaluate results** with respect to acceptability criteria (e.g., compare computed stresses with code and practice requirements).

- **Adjust values of independent variables** as necessary to attempt to satisfy discipline acceptability criteria (e.g., change component materials or geometry to change member stresses). This adjustment represents a local iteration.

- **Present result** to other design team members and client.

These individual activities and their steps continue until the project participants accept a final design and construction plan.

Professionals now typically use software application programs for the data computation, storage, and visualization necessary in pre-construction activities. These software applications support local what-if analyses, but they do not support integrated multi-disciplinary analyses of project design (Pfaffinger 1992). Thus, multi-disciplinary analyses require team meetings and cooperation among discipline experts, a social process that is routine but still slow and frequently incomplete. Progress is now being made toward helping the social process with more rapid and effective AEC software integration. For example, some applications (e.g., Timberline, Primavera, and AutoCAD) now produce file output for sharing data with other applications, and many users find file transfer very valuable. We propose to extend this data sharing capability through theories of organizational and software integration, supported by new software functionalities.

MODELS OF INTEGRATION

This section discusses alternative integration architectures:

- **Organizational integration** is current practice and involves discussions among a client and various discipline specialists and among discipline specialists within project teams or integrated design-build organizations.

- **Technical integration** links software applications that support discipline specialists. It consists of multi-point integration and circle integration.

    - **Multi-point integration** links each application to a central controller that receives all changes made by individual applications and dispatches change notices from an application responsible for the value of a design parameter to other relevant applications.

° **Circle integration** links each application to a single predecessor and a successor. All applications together form a single feedback loop.

These integration architectures are discussed in detail below.

Organizational Integration

Pre-construction design and planning currently involve a number of disciplines, each with responsibility for one or more activities and each often supported by discrete isolated software applications. During project meetings, discipline specialists share results of their individual activities, compare and evaluate results, and modify preliminary design parameter values. These meetings may be small and informal or large and formal. Pre-construction integration is now largely organizational and as systematic as the design manager and design specialists have inclination and time to make it.

Figure 2 shows a simplified representation of the information flow in current practice among pre-construction activities for case example 2. Boxes in the figure name activities. Parentheses enclose the name of the responsible discipline. In each activity, project participants set and compute some parameter values for the project. Arrows indicate data to be passed from one activity to another. Team members responsible for a particular activity decide how to respond to incoming data. Double-headed arrows indicate common feedback loops among the activities of a discipline. Most individual software applications do not provide uniform mechanisms or formalisms for passing data from one application to related applications. Verbal exchanges, paper drawings and reports are frequently used for data transfer between specialists, and often specialists need to take data from memory or paper and enter it into their own software.

Figure 2 illustrates the problem of fragmentation in construction (Howard et al. 1989). Responsibility for design is distributed among multiple specialized entities. Engineering, however, inherently involves use of highly technical knowledge and analysis techniques that now are found in disparate disciplines and organizational entities (the relatively simple days of the master builder of a cathedral having passed long ago). Thus, the architect, structural engineer, etc. all have separate but related responsibilities for the layout of a room, for assuring that it will be built on time, and that it will serve the needs of its intended users.

Organizational integration is intended to integrate the fragmented responsibilities of specialists who each use unique knowledge and techniques to contribute to a total design. Since the software currently in use generally does not support integration, organizations must support feedback among disciplines. Such feedback may be haphazard, depending on how the project manager and the individual discipline specialists communicate, share their recommendations and resolve issues. Because feedback is unstructured, relevant design decisions often get lost or are ignored by busy specialists. As a result, these specialists frequently do not challenge design decisions in a serious and timely way. Lack of quick, serious challenge often hinders the immediate discovery and resolution of (latent) conflicts (see test case 1) and precludes rapid revisions of proposed alternatives.

In current practice, the number of global iterations is usually very small in the pre-construction phase. For instance, the team working in the pre-construction phase of our example university building took nearly a year to develop an initial design. After months of effort of developing a design, there is often little energy or enthusiasm for developing and considering alternative designs, even when clients have requests that are at least partially un-met. The small number of global iterations possible makes it difficult to identify all the important and potentially conflicting design criteria explicitly and to set priorities for them. Thus, the likelihood of improving an initial design is currently low, and the likelihood of finding an optimal solution to given requirements is small.

Current preliminary cost estimation usually builds on high-level summary estimating information, e.g., cost per square foot. It is necessary to apply significant relevant experience to make useful cost estimates early in a project. More detailed cost estimation normally considers detailed construction costs using selected construction methods. Thus, a complete and reliable cost estimate becomes available only at the end of the long series of activities that produces the detailed design and construction plan. In the current, fragmented practice, designers pass on detailed design data to cost estimators, but it is often too late in the process to allow significant design revisions after estimators produce the detailed cost data. The information to evaluate cost in detail might take many months to develop (nearly a year in our first example). An example from industry demonstrates that practitioners have started to use computer tools to help develop accurate cost estimates for complex structures early in the project: architects Frank O. Gehry and

Associates share 3-D models with contractors to develop far more effective cost estimates than had been possible without use of the model (Gehry Forges 1992).

Figure 2 suggests the following set of characteristics of the organizational integration model:

- **Feedback loops are random.** One specialist makes a choice, possibly in direct consultation with other relevant specialists, but possibly with no awareness at all of other relevant specialists. For example, an architect chooses reinforced concrete as the material for a column, consulting with the structural engineer. Other specialists, too, may have interest in the decision, e.g., the fabricator who must cut and bend the reinforcement bars. The contractor who must do the construction may learn of the decision too late to recommend changes in design that would allow the use of construction methods that could significantly reduce project duration. In addition, even when provided with the necessary information, designers often lack enough time for proper engineering review.

- **Feedback is unmanaged.** Partnering and team-building are current organizational methods that facilitate control of design changes and manage feedback. Closer team-oriented partnering has helped to develop higher quality design, but feedback control is still often informal and ad-hoc. For example, decision-making authority and responsibility and the required information are often distributed haphazardly.

- **Feedback is slow.** The frequency of design meetings determines the speed of design feedback. Partly because it often takes days or even weeks to obtain and evaluate design information and partly because participants have multiple responsibilities, design meetings often occur only every several weeks.

- **Significant feedback is very late.** Inconsistencies between dependent variables for one discipline and independent variables for others arise frequently and are often discovered slowly or may be overlooked. For example, cost is usually one of the most important evaluation criteria. Approximate cost estimates are available early in the pre-construction process, based on summary data such as construction cost/square-foot. Detailed cost estimates, however, are not available until all the issues that contribute to cost have been considered. In practice, a good cost estimate is often not available until many months of design have passed. In addition, a design team's inclination to consider alternatives usually decreases rapidly. By the time the detailed cost estimate is available, the design budget is often nearly expended, the inclination to consider alternatives is gone, and the result is that significant feedback is very late, or cursory, or both (Kirby et al. 1989).

- **There are no inherent criteria for defining design versions.** Today, many professionals work on the same project from different locations. In current practice design versions are often defined as the project status at a particular review date. On any given date, however, drawings and specifications tend to be partially incomplete and inconsistent since designers often work with different project data and propose independent and conflicting solutions. With the slow and late feedback noted above, and without a clear definition of what engineering perspectives should be represented in a design version, these conflicts may be detected only during construction or operation. There are no mechanisms in organizational integration that force the analysis by all relevant disciplines and the traversal of local feedback loops.

- **Organizational control may lead to unstable designs.** As in mathematical analysis, stability of solutions is desirable for any process. There are no clear, consistent and well-understood criteria for terminating data propagation from activity to activity. Propagation and hence design may, therefore, be unstable, i.e., even with the same initial client needs, repetition of the pre-construction phase might involve different issues and lead to a different solution.

- **Errors and omissions may occur in moving data from one discipline to another.** Data prepared by one discipline is often misinterpreted by another discipline. Since notes and comments are often hand-written on drawings, they can be missed or misread or might not reach all project participants.

In summary, organizational control provides haphazard, incomplete, and inconsistent project data and specifications that can be shared only slowly and late in the pre-construction process.

Technical integration

As noted above, pre-construction design and planning currently involve various disciplines, each potentially supported by discrete software applications that usually have only limited ability to share data with each other. Technical integration can link these applications so they can pass data among each other. This section discusses issues and architectures of technical integration.

The two most basic architectures for technical integration are automation of the current organizational integration practice and simplification of the data integration. Organizational integration inherently involves people and reflects human behavior, its patterns of data communication will always include randomness, and this opportunistic style is probably appropriate as well as inevitable. There is no need, however, for organizational and technical integration to have the same integration strategies. A rule of thumb in software design is to "simplify the process, then automate."

Integrated software applications can support a "just-what-is-needed" and/or a "just-in-case" strategy of data sharing. Depending on the integration strategy, an integrated system can capture a minimal (hopefully canonical) or an exhaustive subset of project data. Thus, in the simple example of Figure 1, the architect would set values for the column location, material, etc., and the contractor would set the construction method. With pure "just-in-case" integration, all disciplines receive all this information every time they inquire about the current project status; however, with "just-what-is-needed" integration, the architect is not usually informed of the column construction method after the contractor has selected it.

The remainder of this section discusses two architectures for implementing technical integration: (1) multi-point integration implements a completely flexible integration of applications that allows all the rich interactions found in organizational integration; (2) circle integration attempts to simplify the applications' interactions as much as possible.

Multi-point Integration

Multi-point integration links applications in such a way that any application can obtain data from any relevant source and that its outputs are available only to relevant destination applications. Each application broadcasts its results to all listeners, implementing a data sharing strategy with even more arrows than are shown in Figure 2. Figure 3 shows another architecture, one in which each application reports its results to a central controller and any application can obtain any data from the controller. The central controller typically includes at least a database. It will normally also be a knowledge node and will include a control mechanism that receives results from one application and passes them on as input to all other applications the controller finds relevant. It then asks the receiving applications to re-analyze their own outputs given the new input data. Obviously, once one application passes data to the controller, there is no intrinsic way to predict when the effects will stop propagating through the set of applications.

Different technologies can be used to implement multi-point integration, including ad hoc approaches, neutral file exchange, data translation in support of centralized databases (Howard and Rehak 1989), agent-oriented programming (Cutkosky et al. 1993, Khedro et al. 1993), and blackboard systems (Engelmore and Morgan 1988). Abdalla (1989) summarizes some of these integration methods. Froese (1992) discusses use of object oriented tools for integration. Sriram et al. (1989) use object-oriented technology to support concurrent engineering.

Figure 3 suggests a set of characteristics of the multi-point integration model:

- **Feedback loops are implicit.** The multi-point integration model can record all decisions and can pass on all design decisions from the application making a parameter assignment to other applications considered relevant by a central controller, which must also manage feedback.

- **There is no inherent feedback management.** Feedback control is possible if the controller creates and manages feedback, but the controller does not necessarily provide systematic and structured feedback. While inconsistency detection can be added, there is no inherent mechanism for the discovery of inconsistencies between independent and dependent variables in different disciplines.

- **Feedback is rapid.** Linked computer applications can share data rapidly.

- **No inherent criteria for defining design versions exist.** As in organizational integration, no simple criteria specify the disciplines appropriate for any particular version.

- **Designs are potentially unstable.** Like organizational integration, multi-point integration provides no clear, consistent and well-understood criteria for terminating data propagation. Thus, propagation and hence design may be unstable.

- **Data transfer is complete and free of errors.** Assuming correct input data and functioning of the linked applications, all data will be transferred completely and correctly.

The multi-point integration model should facilitate data sharing, with the benefits of rapid feedback. Selective invocation of relatively slow applications may be computationally efficient if the central controller can operate very quickly. Like organizational integration, this model lacks an inherent definition of design versions. In addition, this model requires creating and maintaining knowledge about behavior and capability of each application in the central node.

Circle Integration

This section describes the functioning and use of the "circle" as an architecture for integrating pre-construction activities (see Figure 4). The circle architecture provides a simple mechanism for passing on the parameters set by team members or applications to other applications.

As shown in Figure 4, in the circle architecture applications for each activity have a single predecessor and a single successor. Thus, integration control involves passing information around a circle of applications. Each node receives information, i.e., values for its dependent parameters and parameters in which it has no interest. With some input by discipline specialists, it then sets and computes values for its independent parameters, and passes on all its unchanged input, plus the values it assigned to its independent parameters, to the receiving application. Initially for any run around the circle, a user will enter data at some starting node (often the Layout node for a building design) and initiate analysis and propagation to all subsequent disciplines. In subsequent design iterations, a user can enter changes at any node, and the circle controller will pass on changes until they return to that node and complete a feedback loop.

If each discipline has an independent but identical copy of the integrated set of applications, users for each discipline can initiate the feedback system privately, repeatedly, and independently. In the time required to complete the application on the circle (hopefully at most a few minutes), a discipline specialist can receive feedback regarding the potential effects of a proposed design decision. Then, at design meetings, any discipline specialist can propose design changes, run the integrated system, and the group can discuss both the circle output and the line of reasoning supporting that output. Finally, as in the case of organizational interaction, the group will either reject proposals or consider them as part of a tentative new design version. Accepted proposals constitute a new complete design version. The individual disciplines or the entire team will use this version as the basis for creating the next version. Thus, circle integration allows repeated, managed design iterations during the pre-construction phase. This clear and intuitive notion of version control is a significant advantage of the circle approach over multi-point integration.

An implementation of the circle architecture will propagate the values of all independent and dependent parameters from one activity to the next. When the last node in the circle is completed, the circle outputs a summary of all inputs to the circle and of all outputs from all applications on the circle, i.e., a complete set of project data. Considering client requirements, project participants then evaluate the complete set of data representing one design iteration. Notice that the evaluation now uses complete global data, not only local (single discipline) data. After an evaluation, the system users can accept the tested solution as the "final" solution, or they can propose changes to the independent parameters of any discipline based on their engineering judgment.

As Figure 4 shows, the circle architecture provides a framework for invoking each of the pre-construction activities described above in the Overview Section. The basic idea of the circle architecture is that each application receives propagated data and user input, sets and computes the values of its independent variables, and passes on this output to the next application on the circle. It also passes on the part of the input that it did not modify. In the case of the column example of Figure 2, the preliminary design application receives the column location, height, etc., from the architectural application and then sets the column width, depth, and loads. The preliminary design application passes on the parameter values it received and the values it sets to the next application on the circle. Data that are not dependent variables of a node are forwarded with newly set values to the next downstream activity. For example, the landscape, specified by the architect, is a dependent variable for the estimator but is simply passed on by the earlier structural engineering nodes. For any of a number of reasons, an application locally chooses to ignore or process the individual data it just received. The application then simply confirms that it has received the data and passes them on to the next application. In contrast, the multi-point architecture assumes that a central controller infers what data to send to each application, and it assumes that applications can effectively process all data sent to them.

Because the circle control mechanism is simple and clear, we claim that organizations can manage and maintain it more effectively than they can manage a less structured control mechanism. Circle integration defines an iteration through the circle that is accepted by the complete team as a complete design version. It defines as an individual version an iteration through the circle by a single user. Thus, the circle integration strategy provides a clear and consistent basis for design version management by both individuals and projects.

To enhance efficiency, an activity application could cache the relevant input and output of some number of runs. If a set of input data arrives at an activity and the input matches a set of cached input data, the application would pass the cached output on to the next activity on the circle. Furthermore, a central database can be used as a tool to support the circle architecture. Only a token indicating control needs to be sent from one application to another.

Computational integration currently requires a set of applications, all of which run at the same level of abstraction. Since most AEC application software now works at a relatively detailed level of abstraction, we expect that the initial applications integration will involve detailed abstractions. In addition, an integrated organization still must analyze projects at a level higher or lower than that supported by integrated systems. Research on aggregation and elaboration may lead to applications that work at different levels of abstraction and, later, to integrated systems that change levels of abstraction dynamically as design progresses from concept to details.

Figure 4 suggests a set of characteristics of circle integration. These characteristics become the objectives and suggest the evaluation criteria for an architecture of engineering software integration. These characteristics are:

- **Feedback loops are explicit.** All disciplines review all design decisions, and decisions and implications become both part of the design record. This design record will automatically reveal inconsistencies between dependent and independent variables of various disciplines.

- **Clear feedback loops exist that can be managed simply.** The criterion for terminating feedback is simple: data propagate to all activities in turn, and propagation continues until control returns to the application that initiated a design value revision. Thus, the circle integration provides highly structured feedback that users can control easily.

- **Feedback is rapid.** Complete feedback from all applications is available after each circle run. For example, approximate cost estimates are still available early in the pre-construction process. Detailed cost estimates become available as soon as the user enters all issues that contribute to cost into an application on the circle. With repeated design iterations, cost estimates will become more reliable.

- **Version management is simple and effective.** Each traversal of the circle produces a complete set of project data, i.e., a design version. Over time, design versions will become more consistent, detailed and accurate. Because a design version represents one complete feedback loop, it is likely to be more consistent than design versions are in current practice.

- **Designs are stable.** Unlike in organizational and multi-point integration, there are clear, consistent and well-understood criteria for terminating data propagation. Since circle integration also automatically discovers inconsistencies, design will tend to be more stable than in the absence of circle integration.

Both multi-point and circle integration mechanisms potentially increase the rate of feedback and help overcome shortcomings of organizational integration to make the project participants more effective: multi-point integration adds value to the design process from the perspective of a client by providing feedback more quickly than a strictly organizational alternative does; circle integration offers better feedback management and simplifies version control.

Consider the difference between multi-point and circle integration. In multi-point integration, the controller infers what applications will be affected by a change and what information to distribute to the various applications. While this propagation of data may be computationally efficient, the controller needs to know enough about the pre-construction process to be able to mimic organizational integration adequately and rapidly. Given the current problems of organizational integration concerning data consistency and version management, multi-point integration presents challenging knowledge acquisition and maintenance problems. In addition, the effort to identify, document and maintain the required control paths does not add value to the design process from the perspective of a client. The ongoing documentation effort, while large, leaves the processes of architecture, structural engineering, etc. unchanged. It is an axiom of computer integrated manufacturing that processes should be simplified first, then automated (Hammer 1990).

Circle integration, by contrast, recomputes the entire design and construction plan for every set of changes entered into the system. While this strategy might lead to some re-calculations, it assures data completeness and discovery of consistencies and inconsistencies within each version. Little additional effort must be expended to identify, document and maintain the required control paths. Thus, circle integration adds value to the design process from the perspective of a client by making the process faster and more complete and consistent than the strictly organizational alternative can make it.

To summarize, multi-point and circle integration trade off complexity of development and maintenance with potential computational inefficiency. Given the low cost of fast computers and the high knowledge overhead of multi-point integration, we plan to implement and test circle integration as an integration architecture.

The next sections describe the expected impact of circle integration on engineering practice and discuss implementation issues and related research.

EXPECTED IMPACT OF CIRCLE INTEGRATION ON ENGINEERING PRACTICE

Software that is integrated with a clear integration strategy can help designers to manage design information feedback so that all important design decisions can be documented consistently. When applications share data, design changes proposed by one specialist can be sent to applications used by related specialists for review. Interested experts in related disciplines can then quickly learn about proposed design decisions and provide appropriate critiques to the proposing specialists. Thus, pre-construction integration is an important component of concurrent engineering because it facilitates multi-disciplinary review of design decisions.

In support of the goals of more rapid and effective design, we plan to use a circle integration architecture for the integration of pre-construction software applications to allow designers to perform a complete design iteration very quickly (at most in a few minutes). Each individual designer will have a copy of the complete integrated design system. Figure 5 shows the interaction between team members and software integrated with the circle architecture. In the privacy of their own offices, individual discipline experts can invoke the circle (starting at any node) to identify and explore alternatives. Tentative design decisions can be analyzed in detail by individual discipline analysts between team meetings, and the effects of changes can be analyzed from multiple discipline perspectives. Initial feedback may be based on preliminary or default pre-construction decisions, but this feedback will help the decision-maker(s) to understand implications of these decisions from the perspectives of relevant disciplines. We also expect that the integrated design system will be fast enough to be used interactively during multi-disciplinary design meetings. During a meeting, proposed design choices may be entered and analyzed by the integrated design system, discussed and modified by group members, and after a sufficient number of iterations, accepted by the entire design team.

Successful integration may increase pre-construction costs somewhat as more specialists bill time to this phase of the project. However, software integration should reduce duration of the pre-construction phase, and by bringing more perspectives to the early design, integration should reduce construction and operational costs significantly.

Successful integration will require changes in industrial practice. Participants will need to use the computer more, which may require them to change their work habits. Software integration both enables and requires individual applications to share data. It also enables and requires organizational integration in which specialists share issues and alternatives early and freely. To be successful it will require companies and individuals to pool resources to develop and maintain these integrated systems (Liker et al. 1992). Davidow and Malone (1992) talk about the need for a new high level of trust in professional relationships—surely a change from current practice but one that is emerging in the trend toward design-build and that is required for effective sharing of issues and alternatives.

Potential benefits of successful integration include increased efficiency, reduced time and improved quality for clients.

Early exploration of construction process options will help find constructible designs. As studies in manufacturing have shown, integration of artifact and process design have brought about substantial increases in overall system throughput (Nevins and Whitney 1989), and we anticipate similar benefits in the AEC industry.

Circle integration should lead to order-of-magnitude decreases in the duration of the pre-construction phase. Probably a relatively small number of team meetings (ten to twenty, depending on the size and type of a project) will be required for problem definition and consensus building. A major argument for rapid feedback is that it allows meetings to be held frequently: every few days for a month or two rather than every few weeks for a year. If specialists can do design tradeoff studies in minutes rather than days, meetings can be scheduled every few days. Using integrated tools, individual specialists will be able interactively to do what-if tradeoff analyses, and they will be able to receive immediate feedback regarding the effects of proposed choices. Individual specialists will conduct what-if studies before meetings. When these studies can be done interactively, they may be conducted during a meeting. Major design review meetings may then be held every few days during the pre-construction phase, rather than every few weeks. In addition, when specialists can analyze their choices in preparation for meetings, they can come to meetings with better conceived ideas, and the content and quality of each individual meeting can be increased significantly (Johansen 1988).

Increase in quality will come from the use of integrated tools for two fundamental reasons. First, the customer is an explicit part of the integration, whether the customer is a paying client or an internal next-customer, i.e., the next application, activity, or discipline on the circle. Making customer needs explicit and checking how well they are met can only improve the quality of a project. Second, the number of pre-construction alternatives considered can increase dramatically, and usually consideration of more alternatives leads to better understanding of the design problem and, ultimately, to a solution that is more responsive to customer needs.

These pre-construction benefits will be realized in a number of ways:

Early, structured, precise, and rapid feedback will help specialists to recognize the effects of their choices on other specialties. Complete feedback from the viewpoint of all relevant disciplines is a major benefit of technical integration and will assist discipline experts in their understanding of the impact of their decisions and of the requirements of other disciplines. Knowledge about how particular choices affect individual projects will improve the quality of the choices because decisions will be made only after team members explore multiple perspectives.

Over the course of several projects the state-of-the-practice will be improved because circle integration allows professionals to learn more rapidly through quick exploration of alternatives. Today, it takes several years of practice for a person to become a discipline expert and to understand how his or her particular discipline relates to all the other disciplines. Computer-assisted learning through circle integration is likely to decrease the time it takes to become an expert because for every project, the novice will go through many virtual projects.

We feel that now is the time to implement circle integration because both need and ability exist. The need comes from the current process fragmentation (Liker et al. 1992) and the proliferation of individual software tools. The ability comes from the availability of useful analysis and CAD software and from the very fast, inexpensive desktop computers that are now routinely available to run integrated software applications.

In this paper we focus on horizontal integration of the pre-construction phase. Feedback can be extended from the pre-construction phase to the entire facility life cycle and can include operational effects such as reductions of energy use (Shaviv et al. 1992) and impacts on ease of maintenance.

In the next section we summarize our initial ideas on the implementation of circle integration and on requirements for the testing of such a system.

IMPLEMENTATION AND TESTING STEPS

An implementation of the circle architecture might involve a simple set of linked autonomous applications, in which case the circle output is the union of the outputs of the individual applications. The implementation might also use a central database, and each application could read from and write to the central store.

In this implementation, the circle output would be the state of the central database at the end of an iteration.

To scale up the circle integration architecture from our initial simple set of disciplines, we will need to include additional activities and disciplines involved in the pre-construction phase. The addition of disciplines such as HVAC, electrical, piping, etc., should simply involve adding application nodes to the circle.

Our objective is to test theories of software integration. Careful test cases and test plans will be required to perform and to evaluate tests. The issue of testability immediately suggests the related issue of test criteria. In addition to the criteria identified in the discussion on Circle Integration, different models of technical and organizational integration may potentially be compared in terms of measures of effectiveness in meeting client needs, computation time to develop a single version, and ease of introduction, maintenance and extension. In the absence of controllable test models (e.g., shaking tables for structural engineering, crash tests for automotive engineering, or even market surveys of potential buyers of a new product), selection of relevant evaluation criteria is always difficult.

The following paragraphs suggest several criteria that may be useful in comparing the relative effectiveness of the implementation of different technical integration strategies. Major research effort will be required to develop detailed test evaluation criteria and to perform and evaluate tests using these criteria.

Adaptability: Different project phases require support of different applications. An integration strategy should allow relatively simple addition and deletion of applications.

Clarity of versions: In practice, the integrated system should allow simple and effective creation and management of design versions. Discipline specialists and project managers should understand the contents of design versions and be able to propose relevant changes.

Ease of using commercial software: Industry will want to integrate both commercial and internally developed software applications.

Speed: Since time to create a design and construction alternative is a major objective of integration, it will be useful to observe the time to run a single what-if analysis and to obtain feedback from all potentially relevant disciplines concerning the effects of pre-construction choices.

Stability of solutions: An important question is whether in practice there is a difference in the designs and construction plans produced by similar teams that is attributable to the integration strategy.

Consistency: The independent parameters set by individual disciplines should be shared with all relevant disciplines; each application should interpret dependent data in a semantically appropriate way.

Support for evolutionary development: In practice, research and commercial implementations need to proceed incrementally. Both multi-point and circle integration should allow an initial development followed by incremental addition of both new applications and extensions to the capability of existing applications.

In this paper we have not committed ourselves to an implementation strategy, but clearly the effectiveness of any software integration depends critically upon its implementation. We still need to research appropriate integrated software implementation strategies and goals of integration and automation. In addition to implementation, integration of organizations and software applications brings up a number of research issues. The next section discusses relevant issues.

RELATED WORK AND RESEARCH ISSUES

We drew from previous research in the following areas to develop the ideas presented in this paper: prototype integrated systems, product and process modeling, and the link between organizational design and technical integration. While significant research remains to be done in all of these areas, early results provide evidence of the need and ability for increased software integration to support organizational integration.

The IBDE project, for example, proposed an architecture somewhat similar to the circle (Fenves et al. 1993). IBDE uses a linked set of expert systems to perform individual discipline analyses. It demonstrated that software can assist in rapid design development. It also showed that an important concern of automated tools should be to keep professionals in the loop, evaluating designs. What previous research did not do was research and test effective levels of integration and automation to improve engineering productivity and quality. Other researchers consider related issues in different disciplines. Musen (1992), for example, observes that "the medical-informatics community suffers from a failure to communicate." He uses technical (software) integration to help overcome organizational fragmentation in medicine.

Sanvido et al. (1989) discuss a formalized construction process model that could be facilitated with technical integration, and they describe difficulties in using technical and organizational integration to realize an integrated model. Research on how to formalize, implement, and automate process models is still needed. Gero (1990) discusses AEC design models.

Many researchers discuss product models specialized for representing form data without emphasis on explicit representation of function or behavior. Björk (1989), Gielingh and Tolman (1991), Eastman (1992), Sause et al. (1992) are examples. The product models they propose will contribute immediately to technical integration and, potentially, to organizational integration. To be implemented broadly, these models require a standard (Thissen and Stam 1992), but standards such as PDES/STEP represent data about the form of components only. Circle integration requires a representation that includes form, function, and behavior of objects such as the one advocated by Luth (1992).

One particular research issue concerns levels of abstraction at which different applications around the circle operate. Detailed construction planning and cost estimation, for example, require details that are not available during conceptual building design. One possible solution is to create different versions of applications for each discipline, one for each supported level of abstraction. Circle users might work at one level of abstraction and then, when data become sufficiently precise, the design team members could invoke all applications at a more detailed level of abstraction. Tauber et al. (1992) use the idea of a "defending champion" to address this issue: they relate a conceptual design to a selected detailed design taken from a library of case examples. When detailed analyses are needed, an analysis program uses the details of the defending champion. The circle design process incrementally replaces the details of the defending champion with the details of the emerging design.

Ettlie and Reza (1992) argue that successful organizations use two types of mechanisms to capture value from process integration: the first uses process integration as an occasion for significant restructuring; the second uses integration to create new business practices. They claim that successful organizations use a combination of organizational integration and process innovation that is facilitated by technology. The question of how to combine technical and organizational integration to generate competitive advantage is still a research issue, however.

Technical and organizational integration clearly interact. Computer-based models of organizational behavior have started to emerge (Masuch and LaPotin 1989, Carley et al. 1990, Cohen 1992). Technical integration can be added as a tool for organizations in these organizational models. Synthetic experiments can then be used to study the interaction between organizational and technical integration. Studies of technical and organizational integration can address issues such as management of the time lag between identifying and resolving issues, accommodation of legal and traditional organizational practices, incremental introduction into industrial practice, and change of responsibility for independent variables.

Another issue concerns the role of control. A project manager manages organizational integration, in multi-point integration, a central agent manages the information and control flow, while the circle lacks an explicit central control agent. It is currently not clear whether the central agent can be implemented and maintained economically and whether the simple control of the circle supports all team members effectively.

Scale-up is another major issue: only limited tests can be performed in a research lab, yet we hope to develop guidelines for scale-up to support commercial complexity.

This paper considers the pre-construction phase of engineering. Relatively obvious extensions will support construction, and operation and maintenance. Construction will include construction monitoring and control and, eventually, robotic construction. Technical integration will support operation and maintenance by monitoring system status such as on-line energy and safety, and suggesting maintenance actions (Jin et al. 1992). As the phases become integrated, information from design can be fed forward to construction and operation and maintenance systems, and information from construction and operation can be fed back to improve future designs.

SUMMARY

We offer this discussion as a rather pure model to suggest testable hypotheses about AEC software integration design. We assume that any practitioner will find our examples simplistic, and we expect that some (perhaps significant) extension of detail will be required for a commercial implementation of software integration. The illustrative examples suggest important issues and opportunities for integrating the current generation of individual AEC software applications. The objective in this discussion is to suggest ideas that can be implemented and tested with limited resources. Our own long-term research objective is to demonstrate and actually use realistic industry data to test the technical viability and operational effectiveness of alternative theories of software integration. We hope that many other academic and industrial investigators will join this effort to develop and test integration strategies. During this testing, we will identify the assumptions that must be made to make software integration feasible in practice and learn how difficult these assumptions are to satisfy in practice.

Organizational integration has emerged in practice in response to real needs for both large amounts of detail and management of a complex process. Thus, in practice, we expect to see combinations of technical and organizational integration that become increasingly complementary. Integration will not substitute for the ingenuity of project participants, but it will help bring order to chaos.

As Dyson suggests (1992), integrated teams and software will support concurrent engineering and will provide a more flexible response to changing client needs and handling of exceptions during design, construction, operation and maintenance. Integration provides the framework for organizing disparate project activities, the mechanism for feedback among activities, and the specification of the need for sharing data. Automation provides the engine to drive this integration mechanism. Integrated project teams will continue to relate to clients and to use engineering and business judgment to manage the engineering process in general and the integration in particular.

In the university building case example, the first complete detailed design version became available after about eight months. Following initial data entry, the proposed technical integration will compute (potentially inconsistent) detailed design versions in a few minutes. The eight months to a few minutes ratio is roughly five orders of magnitude. The price of reaching for such a dramatic increase in productivity is only a number of years of research.

ACKNOWLEDGMENTS

We would like to thank our colleagues and students who provided input to and commented on earlier drafts of this paper, specifically including James Breuer, Tore Christiansen, Renate Fruchter, Yan Jin, Taha Khedro, Kincho Law, Ray Levitt, Bill O'Brien, Boyd Paulson, Douglas Phan, and Paul Teicholz.

REFERENCES

Abdalla, G.A., (1989). *Object-Oriented Principles and Techniques for Computer Integrated Design*, PhD Thesis, Dept. of Civ. Engrg., Univ. of Calif., Berkeley.

Björk, B.C. (1989). "Basic Structure of a Proposed Building Product Model," *Computer Aided Design*, 21(2), 71-78.

Carley, K.M.C., Kjaer-Hansen, J., Prietula, M., and Newell, A. (1990). "Plural-Soar: A Prolegomenon to Artificial Agents and Organizational Behavior," Working Paper, Center for Management of Technology, GSIA, CMU.

Cohen, G. (1992). *The Virtual Design Team: An Information Processing Model of Design Team Management*, PhD Thesis, Dept. of Civ. Engrg., Stanford Univ.

Cutkosky, M.R., Engelmore, R.S., Fikes, R.E., Genesereth, M.R., Gruber, T.A., Mark, W.S., Tenenbaum, J.M., and Weber, J.C. (1993). "PACT: An Experiment in Integrating Concurrent Engineering Systems", Computer, 26(1), 28-38.

Davidow, W., and Malone, M. (1992). *The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century,* Harper Business.

Dyson, E., (1992). "Workflow," *Forbes*, 11/28, p. 192.

Eastman C.M. (1992). "Modeling of buildings: evolution and concepts," *Automation in Constr.*, 1(2), 99-109.

Engelmore, R., and Morgan, T. (1988). *Blackboard Systems*, Addison-Wesley.

Ettlie, J.E., and Reza, E.M. (1992). "Organizational Integration and Process Integration," *Acad. of Mgt. J.*, 35(4), 795-827.

Fenves, S.J., Flemming, U., Hendrickson, C., Terk, M., Woodbury, R.F., Maher, M.L., and Quadrel, R., (1993). *Concurrent Computer-Integrated Building Design*, Prentice Hall.

Froese, T.M. (1992). *Integrated Computer-Aided Project Management Through Standard Object-Oriented Models*, PhD Thesis, Dept. of Civ. Engrg., Stanford Univ.

"Gehry Forges New Computer Links: Aerospace-developed software translates curved forms into crafted construction," *Architecture*, August 1992, 105-110.

Gielingh, W., and Tolman, F. (1991). "Information Integration in the Building and Construction Industries," *Microcomp. in Civ. Engrg.*, 6, 329-334.

Gero, J. (1990). "Design Prototypes: A Knowledge Representation Schema for Design," *AI Mag.*, 11(4), 26-36.

Hammer, M. (1990). "Reengineering Work: Don't Automate, Obliterate." *HBR*, 68(4), 104-112.

Howard, H.C., and Rehak, D.R., (1989). "KADBASE: A Prototype Expert System-Database Interface for Engineering," *IEEE Expert*, 4(3), 65-76.

Howard, H.C., Levitt, R.E., Paulson, B.C., Pohl, J.G., and Tatum, C.B. (1989). "Computer Integration: Reducing Fragmentation in AEC Industry," *J. of Comp. in Civ. Engrg.*, ASCE, 3(1), 18-32.

Huber, G.P. (1990). "A Theory of the Effects of Advanced Information Technologies on Organizational Design, Intelligence, and Decision Making." *Acad. of Mgt. Rev.*, 15(1), 47-71.

Jin Y., Kunz, J., Levitt, R., and Winstanley, G. (1992). "Design of Project Plans From Fundamental Knowledge of Engineered Systems," *Proc. AAAI 1992 Fall Symposium: Design from Physical Principles*, Cambridge, MA, 149-154.

Johansen, R. (1988). *Groupware: Computer support for business teams*, Free Press, New York.

Khedro, T., Genesereth, M.R., and Teicholz, P.M. (1993). "Agent-Based Framework for Integrated Facility Engineering," *J. for Engrg. with Comp.*, in press.

Kirby, J., Cannaite, R., Hicks, D., and Japel, E. (1989). "Constructibility and Design Reviews: Analysis and Recommendations for Improvement," *USACERL Tech Rep*, P-89/15.

Liker, J.K., Fleischer, M., and Arnsdorf, D. (1992). "Fulfilling the Promises of CAD." *Sloan Mgt. Rev.*, Spring, 74-86.

Luth, G., (1992). *Representation and Reasoning for Integrated Structural Design*, PhD Thesis, Dept. of Civ. Engrg., Stanford Univ.

Masuch, M., and LaPotin, P. (1989). "Beyond Garbage Cans: An AI Model of Organizational Choice," *Admin. Science Quart.*, 34, 38-67.

Musen, M.A. (1992). "Dimensions of Knowledge Sharing and Reuse," *Computers and Biomedical Research*, 25, 433-467.

Nevins, J.L., and Whitney, D.E., Eds. (1989). *Concurrent Design of Products and Processes: A Strategy for the Next Generation in Manufacturing*, McGraw Hill.

Pfaffinger, D.D. (1992). "Recent Advances in Data Exchange in Structural Engineering," *Struct. Engrg. Intl.*, 273-274.

Sanvido, V.E., Kumara, S., and Ham, I., (1989). "A Top Down Approach to Integrating the Building Process," *J. for Engrg. with Comp.*, 5, 91-103.

Sause, R., Martini, K., and Powell, G. (1992). "Object-oriented Approaches for Integrated Engineering Design Systems," *J. Comput. Civ. Eng.*, 6(3), 248-265.

Shaviv, E., Kalay, Y.E., and Peleg, U.J. (1992). "An integrated knowledge-based and procedural system for the design of energy conscious buildings," *Automation in Constr.*, 1(2), 123-142.

Sriram, D., Stephanopoulos, G., Logcher, R., Gossard, D., Groleau, N., Serrano, D., and Navinchandra, D. (1989). "Knowledge-based System Applications in Engineering Design: Research at MIT," *AI Mag.*, 10(3), 79-96.

Tauber, E.R., Levitt, R.E., Oralkan, G.A., Reinberg, F.C., and Walsh, T.J. (1992). "The Conceptimator: An Expert System for Conceptual Cost Estimating of Building Foundations," *CIFE Working Paper*, 14, Stanford Univ.

Thissen, W.A.H., and Stam, W.J. (1992). "Electronic data interchange in an industrial sector: The case of The Netherlands' building industry." *Inf. & Mgt.*, 23, 15-30.

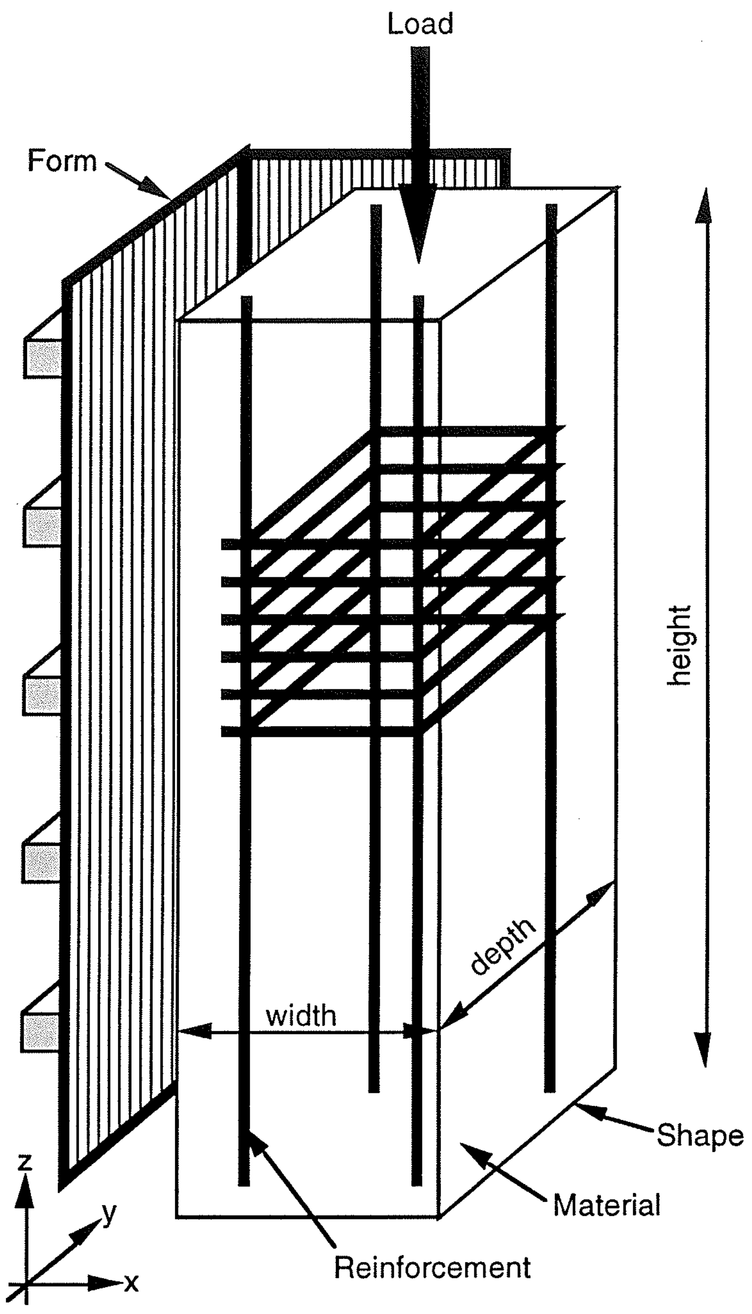

Figure 1. Case example: A reinforced concrete column. Pre-construction activities specify values of parameters of components and systems. Objectives of integration are to distribute the parameter values set by one discipline quickly and consistently to other disciplines to allow rapid evaluation of alternatives.

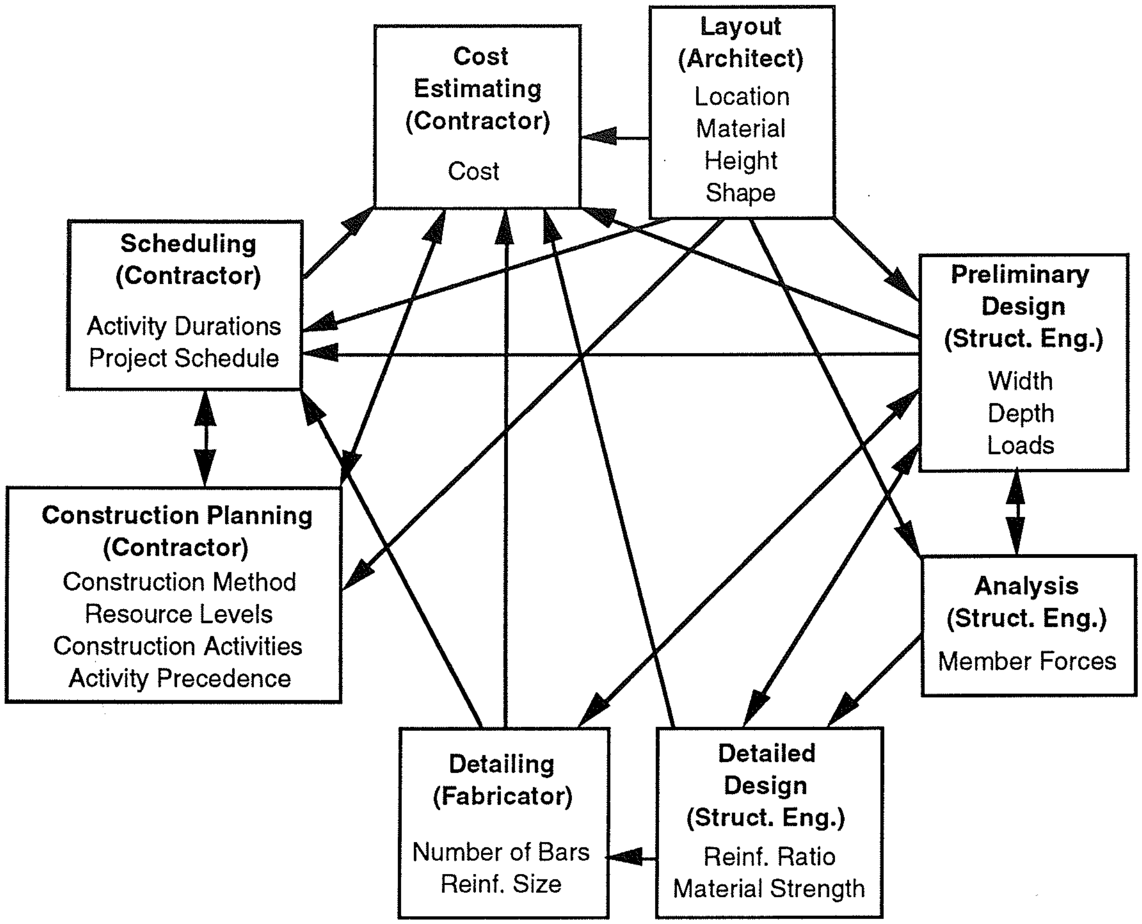

Figure 2. Current Integration: organizational, with some activities aided by software tools. The complexity of inter-disciplinary interaction often impedes effective sharing of design decisions across disciplines. Arrows indicate typical information exchange between disciplines.

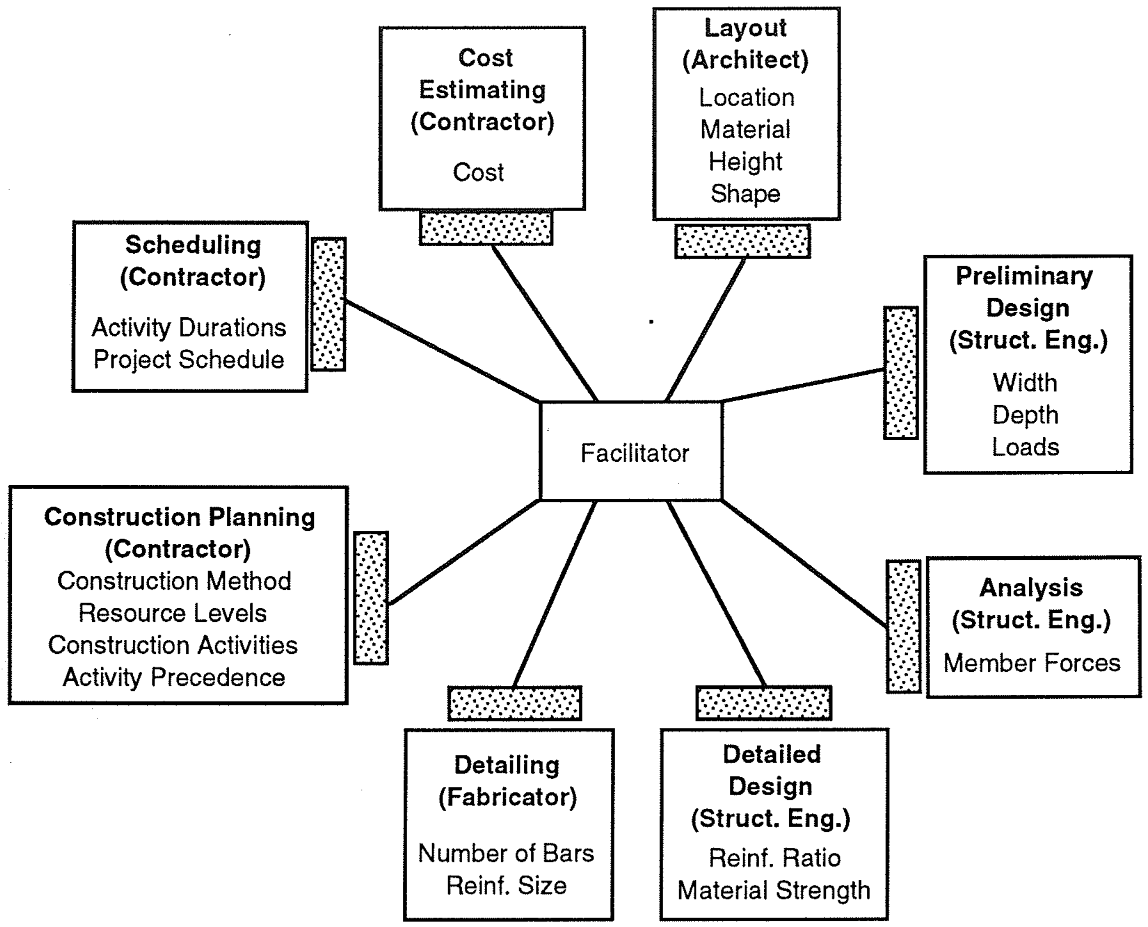

Figure 3. Multi-point integration. One possible implementation of integration of pre-construction software involves linking each application to a central controller that acts like an "intelligent switch." The small stippled boxes indicate the communication routines added to all application nodes to allow them to send data to and receive data from a central controller.

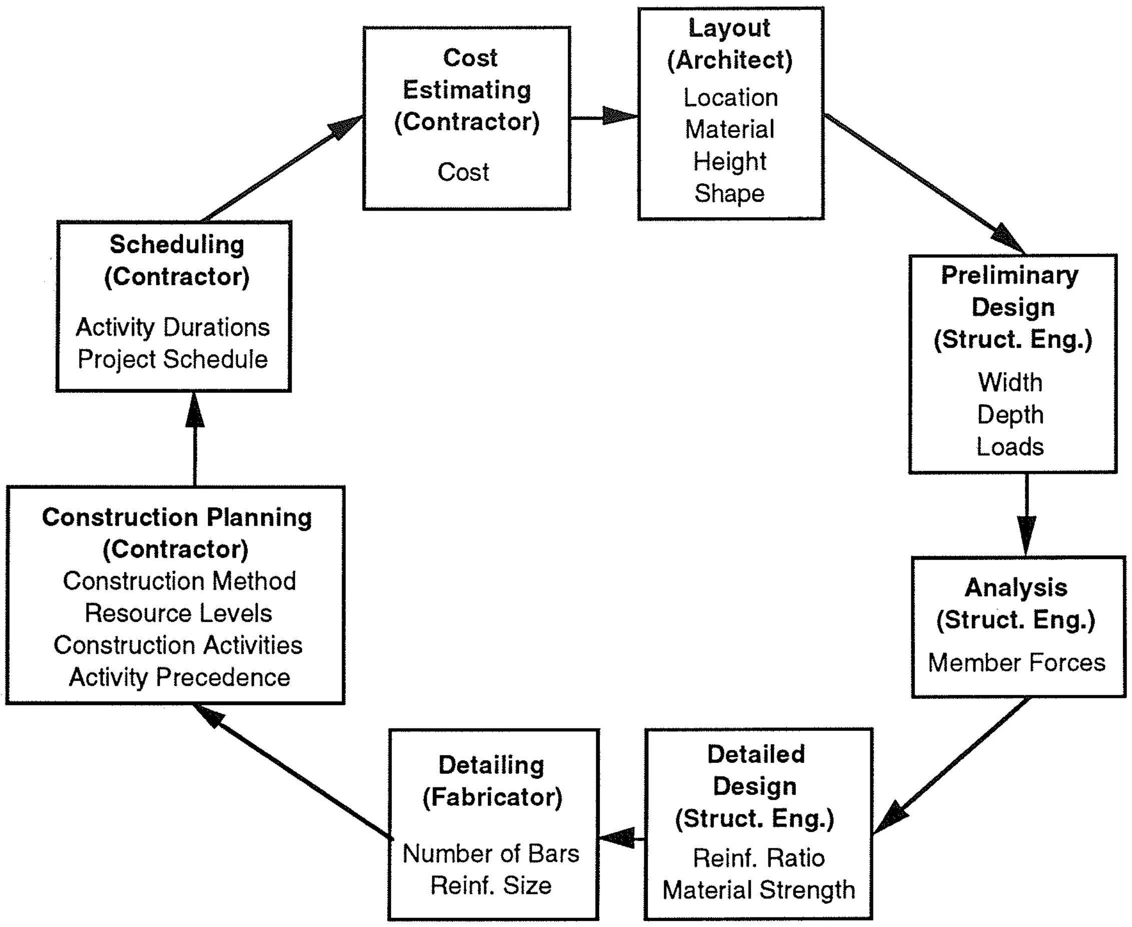

Figure 4. Circle Integration: Going around the circle. Each activity in this figure creates values of parameters and passes those values on to the next activity (along the path indicated by arrows). A single pass around the circle constitutes one design iteration. Ideally, each pass around the circle will require only a few minutes. Users responsible for individual activities can review the entire design, including the parameters for their own activities, and suggest changes to their individual design activities to help local and global optimization.

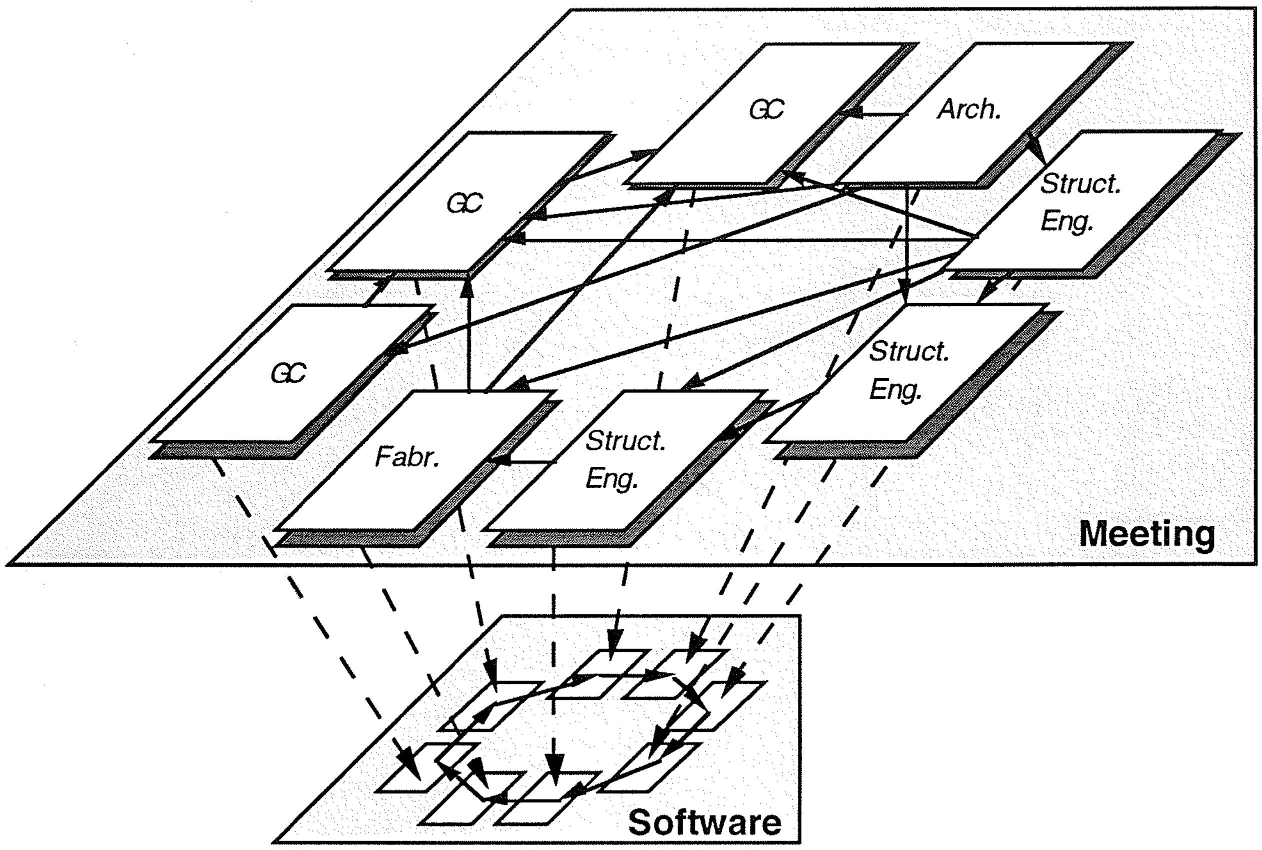

Figure 5. Integrating organizational and circle integration. Circle integration supports individual project participants and the entire project team. It also provides consistent feedback. The users close the feedback loop.

	Layout	Preliminary Design	Analysis	Detailed Design	Detailing	Construction Planning	Scheduling	Cost Estimating
Variable	Architect	Struct. Eng.	Struct. Eng.	Struct. Eng.	Fabricator	Contractor	Contractor	Contractor
Column Location	X							
Column Material	X							
Column Height	X							
Column Shape	X							
Column Width		X						
Column Depth		X						
Loads		X						
Member Forces			X					
Reinforcement Ratio				X				
Material Strength				X				
Number of Bars					X			
Reinforcement Size					X			
Construction Method						X		
Resource Levels						X		
Construction Activities						X		
Activity Precedence						X		
Activity Durations							X	
Project Schedule							X	
Cost								X

Table 1. Summary of important disciplines involved in pre-construction activities (columns) and examples of their variables (rows) for case example 2. Table entries indicate the usual discipline that sets and computes independent variable values. Most variables become inputs (dependent variables) to the other disciplines that do not set them.