Constraint Technology: Not Ready for Prime Time?

David Kurlander - Microsoft Research

Though constraint-based systems have existed for over thirty years, they have never made it big in the software market. Here I will discuss why this has been the case, and what must be done to make constraint technology appeal to the mass audience. At Microsoft Research, our charter is to not only develop useful technologies in house, but also to find promising technologies being developed elsewhere, and bring them to the attention of our product groups. Right now there are a number of obstacles that must be overcome before we start putting constraints in our products.

Almost all of Microsoft's products are geared towards the mass market, and our mass market products sell in the hundreds of thousands or millions. Constraints have been featured in niche products from other companies, but there have been few or no major commercial successes for constraints. Now why is this? For a product to sell in the millions, it has got to be cheap. When the product is cheap, the software vendor cannot afford to give individual attention to each customer. Right now, users find constraints difficult to work with. I'll explain the many reasons for this momentarily, but unfortunately, users of popular products have to both train themselves, and solve their own problems. They have to be extremely self-sufficient, and provide their own technical support. Further complicating things is that the user base for mass market products includes people with little formal education and technical computer knowledge.

Now there is really a continuum between mass market products and niche products, and a product does not have to sell in the millions to be successful. There have been successes in the CAD industry for constraints, but CAD programs are expensive, CAD users tend to be highly skilled, and they tend to get a higher level of support from their software providers.

I do not want to sound too negative. Microsoft is very interested in constraint technology for a number of applications. First, several of our applications have drawing components. In the future, these disjoint components will be unified, and we would like the drawing editor to be considerably more capable. By incorporating constraints in the editor, we would allow people to create more precisely dimensioned illustrations, and make it easier for people to change a single component of the illustration and have the other components reconfigure as desired.

Microsoft is also building a multimedia authoring environment. We would like to provide constraints to allow the author to specify how objects move in relation to each other, and possibly for temporal sequencing. Recently Microsoft bought SoftImage, which is one of the major vendors of 3D modeling and animation software. There is also interest in adding constraints to their products.

Now the interest from these groups is very tentative, and I wouldn't be surprised if they chose to ignore constraint technology for a while. All three of these groups are interested in using graphical constraints, and there are a number of difficulties that people face when using these constraints.

First, graphical constraints can be hard to specify. It can be very difficult to come up with all the necessary constraints without a great deal of trial and error. When the constraint solver is invoked with a partially specified system, drawings often become degenerate, with vertices unexpectedly becoming coincident or collinear. Also, constraints are often at the wrong abstraction level. For example, the user might think in terms of the desired motion of a mechanism, and not fixed location, fixed distance, and relative rotation constraints.

Second, constraint systems are notoriously difficult to debug. When multiple constraints conflict, there should be a way of indicating which are the problem constraints. Typically there is not. Also it is a challenge to design a visual representation for constraints that does not overly clutter the scene, and is intuitive to the user. A good visual representation is essential to debugging as well as to initial specification.

Finally, graphical constraints become significantly slower to solve as the systems become larger. Furthermore, in some cases, the addition of a single constraint can dramatically slow down the system. We need to develop techniques for accelerating the solving process, and estimating to the user how long the solution might take.

There are two approaches that show promise in reducing the difficulty of declaring constraints and debugging them. We can build systems that automatically infer the constraints that the end user intends, thus simplifying the specification task. We can also encapsulate low-level constraint specifications in higher-level constraint templates. This latter technique simplifies constraint specifications by allowing sets of constraints to be predefined and debugged by experts, and bound to higher-level abstractions.

Actually, one of the most important reasons that we don't see constraints in mass-market products is that few people in industry know much about them. Fairly frequently, I am asked about constraints by people in product groups, but there is no good book that I can point them to. Furthermore, it is rare to find a college computer science course that has more than a single lecture on constraints, and even a single lecture is rare. New constraint practitioners have to learn about the field by collecting and reading an assortment of papers. I asked one of the managers of our multimedia authoring effort why constraints are not in the current architecture. He responded that he hopes to add them, but frankly, nobody in his group has the necessary expertise to design them into a system or implement them. One of the things that would best foster the propagation of constraints into industry would be an excellent textbook, and university courses devoted to the subject.

There are a number of trends that I believe will encourage product developers to take constraints a little more seriously. Faster machines allow users to apply constraints to larger problems, though the algorithmic complexity of most numeric solvers will continue to be somewhat prohibitive. Mass market applications these days are getting better and better, and with each release, developers try to differentiate their products from their competitor's by adding more and more features. Sooner or later, constraints will make it onto the list of features that certain applications must have to be taken seriously. These days, product cycles are getting shorter. This, together with faster computers, will encourage people to use constraint-based development environments.

Perhaps I have been too negative in my assessment of how constraints have faired commercially. Spreadsheet programs and project management software feature very simple constraint capabilities, and have been successful in the marketplace. These programs succeed because they present a relatively straightforward interface to the user, and limitations to the constraint model prevent complexities that would be present in more sophisticated constraint systems.

Such applications suggest that the secret to building successful constraint-based systems is to focus on usability issues. Constraints should be easy to specify and debug. They must have understandable visual representations, and be at an appropriate level of abstraction. They must produce predictable results in a predictable amount of time. As I mentioned before, I believe constraint inferencing and complexity encapsulation to be important for the adoption of graphical constraints in many applications, and their mass market acceptance. Both of these techniques are essentially user interface approaches for dealing with constraint specification. Better constraint coverage in computer science programs will also play a role in enabling more software engineers to incorporate constraints in their systems. I encourage constraint researchers and practitioners to constantly be thinking of the big picture: what can be done to make their work useful, and accessible to large numbers of people. And hopefully then, constraint technology will in fact be ready for prime time.