COMMERCIAL SOFTWARE DEVELOPERS VIEWPOINT

Automation of draughting and building modelling - historical review of commercial development since the seventies.

Paul Richens

McDonnell Douglas
Wellington House
East Road
Cambridge
England

October 1988

KEYWORDS

Architecture, Assembly model, BDS, Building, Design, Drafting, GDS modelling, Solid model, User interface, Visualisation

ABSTRACT

The present day GDS system has its roots in BDS, started in 1970, BDS was a 3D data-centred system for design, analysis and documentation of system-built buildings. GDS started as a 2D drafting system, and proved more effective and marketable. Specialised applications and 3D capabilities were added gradually. Current interest is in simplifying the software, especially its user interface.

INTRODUCTION

I am going to talk today about our experiences in applying computer graphics and geometrical modelling to architectural design and the construction industry. It is the story of growth - growth of a software product called GDS, growth of a business, and growth of our own experience and skill.

The story starts for us in the early seventies, when a group of researchers at the School of Architecture in Cambridge banded together to form Applied Research of Cambridge Ltd[1], a commercial organisation intended to exploit their research.

At that time there was a major boom in public building (schools, hospitals and housing), and a serious concern that the construction industry could not cope with the load. There were many initiatives to improve design and construction such as modular coordination, prefabrication, system building and quantitative methods for design and analysis. In this idealistic environment we acquired our first clients, who were public sector bodies interested in the whole life cycle of their buildings. They took a long term view, and were prepared to spend extra money on the design (and even on the construction) of buildings if they could thereby reduce the cost-in-use.

So started the first phase of our history, which was one of Idealism. It lasted from 1970 to 1978, and gave us a mere handful of clients. It did not last; by 1979 we entered our second phase of Realism, and by 1982 had acquired around 50 customers. Our next age was one of Pragmatism and saw a very substantial growth in the business, to around 700 customers. That brings us to the present; I think now we are looking for a new philosophy, perhaps Minimalism, which will help us expand to reach thousands of customers.

This paper will describe these four stages of our development and the shifts in philosophy that occurred with our increasing success and understanding of the market.

PHASE 1 IDEALISM (1970-1978)

Our greatest success in this period was the development of OXSYS, later renamed BDS. Our client was the Oxford Regional Health Authority (ORHA); they had a large programme of hospital buildings and had already developed a system of prefabricated construction called the Oxford Method. Now they wanted a complementary software system for design and documentation.

---

1. Acquired by McDonnell Douglas in 1985

This was our age of idealism; we set out to implement a centralised design database accessible to all disciplines, recording all design decisions, supporting all kinds of analysis and design automation. Finally it would produce coordinated drawings and schedules, which being derived from the central database, would have none of the inconsistencies which usually bedevil a large project.

The design process as idealised by our client was:-

a) Establish brief (i.e accommodation and equipment needs)
b) Devise 3 or 4 strategic alternative building forms
c) Evaluate alternatives and select one
d) Refine detail planning for selected alternatives
e) Develop construction details
f) Issue construction documentation

This is a more straightforward and linear process than we usually encounter, because the materials and components of construction are pre-defined and held constant by the Oxford Method. They are not design variables. During evaluation we could make a lot of assumptions about the physical properties and cost of the building, and therefore perform sophisticated analysis without a lot of input. Detailing, it was anticipated, could be largely automated by using software incorporating the rules of assembly of Oxford Method. Final drawings would be produced automatically from the developed model.

These objectives were largely achieved. The system was built around a design database containing two models; the planning model and the detail model.

The planning model represented the building as a set of nested polygonal extrusions. The outermost polygons represented the building envelope, inner ones floors, departments and rooms. This model was built and analysed during the conceptual design stage. It had sufficient detail to support cost planning, thermal analysis, ventilation and traffic flow predictions. Sketch plans and block perspectives could be extracted from it.

The second part of the model dealt with detailing. First a library of components was created, to define the Oxford Method. Each component had attributes describing its coordinating dimensions, plan and elevation views, cost and so on. Then the design was detailed by recording the locations of these components in the planning model. Sometimes (e.g ductwork) we also recorded their interconnectivity. When every component needed to build the building had been located, the design was complete, and we could extract the plans and schedules required for construction. Simple!

Software was developed for detailing each element (structure, floors, cladding, partitions, services and so on). This software embedded the "rules of assembly" of Oxford Method. If we had been writing it ten years later we would doubtless have externalised the rules in a "knowledge base" and called it an expert detailing system.

This software performed fairly well at ORHA. But problems arose when we tried to sell it to a wider market.

PHASE II REALISM (1979-1982)

The idea of a central complete building model supporting all design functions had (and still has) terrific market appeal. So we soon had a few customers trying to use BDS on a diversity of building projects. These projects used traditional construction rather than system building, and the apparent advantages of BDS crumbled, for a number of reasons:-

1. The linear design process does not apply when there is a choice of construction as well as of building form.

2. The normal design sequence tackles concept first and details later. BDS requires the component library (containing details) to be created first. This is difficult and unnatural.

3. The pay-offs of easy analysis and automatic detailing were not available outside Oxford Method. We were left with a clumsy method of producing drawings.

The age of realism dawned when we perceived that:

a) BDS was being used almost entirely to produce working drawings.

b) The BDS model contained more information than appeared in the drawing set and that model building was actually and theoretically an inefficient way of producing the drawings.

We argued that the main output of an architects office consists of drawings, that about 40% of their working hours is devoted to drawing, and that automation would have the greatest impact if it focussed directly on drawing production.

Our next product was GDS, a drafting system aimed directly at drawing production. It was determinedly two-dimensional, and made no attempt to provide a central single design model. Multiple representation was actively encouraged.

In these respects it contradicted our idealism in BDS. We retained, however, the idea of a component - called an object in GDS. All graphics and text belongs to an object, which has a name. This simple but mandatory structuring of drawn information remains an important differentiator of GDS.

On the functional side we provided a very full set of construction functions for lines and circles, and comprehensive editing, so that anything you did could be readily altered or undone. Finally we paid very great attention to graphical quality.

GDS arrived on the marked at just the right time. It could compete easily with BDS and similar systems (such as RUCAPS) because of its flexibility - it could produce many more types of drawings and was far more efficient. It competed well with systems from Intergraph and CV because of its quality, object structure and roots in architecture. Our business began to expand and by 1982 we had around 50 users.

The first important addition to GDS was the "Attribute Data" sub-system. This allowed properties (text or numbers) to be attached to objects, or instances of objects. The second was the XBASIC customer programming language. This was a Basic interpreter with extensions for manipulating the GDS geometry and attributes, for drawing, and for user interaction in the GDS style. These two additions opened enormous opportunities for extending and specialising the functionality of GDS.

PHASE III PRAGMATISM (1983-1987)

The next phase was lively and exciting. We had a growing group of customers, and all sorts of people in different design disciplines (some very remote from architecture) began to approach us. We started to look for opportunities to develop and add to the basic GDS capabilities so as to expand our market and our usefulness.

Our general aim was to expand from the architectural market to related building design discipline. Our first venture was into structural engineering with a Reinforced Concrete Detailing System. It was quite easy to utilise the GDS object structure and attribute data capabilities to represent reinforcement details as drawings, and to extract bending schedules from them. We did this by adding functions to GDS; no extra structures or concepts were needed.

The second was a little more elaborate, and was for modelling ground surfaces. Land survey data is captured by GDS, and recorded as spot-heights at random locations. The site system reads this data from a GDS drawing, and builds a triangulated representation.

This can be contoured, measured, modified and presented in a number of ways, and the results returned to GDS. The site system added no concepts to GDS, but does use its own specialised, temporary data structures while it is working.

During this period the rate of development of hardware, both processors and graphics displays, began to accelerate. An increasing part of our development effort had to be dedicated to keeping up-to-date. GDS originally used storage tube displays, keyboard commands, 16-bit processors and an overlayed program structure. Over a couple of years we moved to raster graphics, a tablet interface, 32 bit processors and virtual memory.

The determinedly 2D outlook of GDS had forced a break with the "whole building" modelling of BDS, and had enabled us to produce something less ambitious but more complete (in its own terms) and far more effective in practice. But it was not a sustainable philosophy in the face of the rising demands of our customers, and strengthening competition from other vendors.

We began to add 3D capabilities, taking advantage of raster graphics and virtual memory, but without restoring the rigidities of BDS. We felt that 2D approaches were best for most routine work, 3D should be used in special cases, in limited areas. We aimed to get a smooth mix of 2D and 3D capabilities.

We developed 3 subsystems, XSOLID for creating solid models, PAM (Part Assembly Model) for locating them in space, and SVS (Super View System) for rendering.

The job of XSOLID is to create 3-dimensional objects. Contemporary architectural systems were all less than fully functional in this respect. BDS and RUCAPS used the gross approximation of "box geometry". Others offered simple 3D shapes such as wirelines, planes, simple extrusions and surfaces of revolution. We determined to borrow true solid modelling technology from the world of industrial design and mechanical engineering. The idea is to start with simple shapes such as extrusions or solids of revolution, and then combine them, or use one to cut holes in another, so as to create shapes of arbitrary complexity.

The Part Assembly Model uses a GDS 2D drawing with attribute data to describe the positions of 3D components. They appear in the appropriate projection on the PAM drawing, and may be mixed with 2D objects. A PAM plan can be processed automatically into another view such as a section or elevation. This gives us a fundamentally 2D interface for defining 3D models, which is simple to use, and costs less in computer resources.

The final module SVS, will take 3D data (from a PAM drawing or elsewhere) and render it as a perspective projection. We offer a number of rendering algorithms for different purposes; so you can trade-off between the sophistication of the image and the time it takes to generate. The best images use expensive ray-tracing techniques (e.g for shadow casting and reflections), and need 24 bit colour displays. At present these are rather too expensive for routine use in architecture, so we have spent a good deal of effort on faster algorithms, and techniques for "dithering" colours to get acceptable images on terminals with a lower colour resolution.

Our ambition to offer something useful to all disciplines involved in building design has now broadened to cover not only buildings but all kinds of infrastructure. So now we do a lot of work in roads, transit systems and mapping. One problem that we faced and solved for mapping has had a tremendous pay-off for architects and buildings designers. A mapping database (e.g for local government) is likely to be very extensive in area, and carry data from many departments (e.g highways, land tax, drainage, building permits). Many people will be using it at the same time, so concurrency control becomes an important issue. Originally a GDS session would open a single drawing file, and the normal locking protocols apply - many people can read a file simultaneously, or alternatively one person can update it. To solve the mapping problem we did two things. A session can now access many files at once. These can be regarded as "layers" seen in superposition, or as a mosaic of drawings seen in juxtaposition, or both. The boundaries are invisible, and files are opened and closed automatically. So our user can access a large database consisting of thousands of maps as though it were one seamless drawing. The second change was to the concurrency protocol, to permit many readers and one writer to coexist. In combination these two changes allow for effective multiple access to a large mapping database. Update activity does not lock-out readers, and only inhibits other updates if they are to precisely the same layer of the same small region.

The use of multiple layers has proved invaluable in building design, especially in multi-disciplinary offices. Each discipline has its own layer, and the exclusive right to update it. Other disciplines can see it, but not change it.

Our pragmatic phase greatly extended the amount of software we had on offer, the size of our business, and the number of users. By 1987 we had about 700 users, and world-wide revenues of between $40 million and $50 million.

PHASE IV MINIMALISM 1988 ....

We continue of course to expand the software - we are currently working, for example, on land-use polygons for administrative mapping, road design, and extension to the 3D systems for architecture and plant. But I do not see these as the major issues facing us at present. The real issue is to simplify what we have.

The expansion of the last period has left us with over a million lines of code, 800-odd commands, and 11,000 pages of documentation making up a metre of manuals. The rigidities of BDS, or the first period of GDS have gone; we have in effect about a dozen loosely coupled systems with rather less consistency than I would like. The result is getting to be rather expensive to maintain, to learn or to use.

We would like to expand our user base into the thousands, and increase the penetration, that is the number of people in each user organisation that use our software. To do this we have to a) reduce the cost and complexity of the hardware, and b) to simplify and consolidate the software.

The most exciting work we are doing at present towards these aims are concerned with the user interface. By employing the X-11 software from MIT, running on the latest networked engineering workstations, we can develop a direct-manipulation user interface similar to that on the Apple Macintosh. We hope thereby to decrease the documentation load, shorten the learning curve and make the software vastly more "approachable". An unexpected, but very welcome, benefit of this style of interface is that it enables us to considerably reduce the number of separate commands or functions that the user sees. In some cases we can replace a dozen old style commands by a single new style contral panel, and simultaneously make it more obvious what the options are.

Our early work was very much "data-centred". BDS tried to develop a central database capable of supporting a wide range of activities. Later work was "output-centred". We looked at what output was required, and designed software to achieve it in the most cost efficient way.

Our current pre-occuption with simplicity leads me to suggest that the important thing is to minimise the number of concepts in a software system. Concepts should be few in number, simple to grasp, limited in their interactions, and continuously and consistently reinforced by the user interface. These principles apply regardless of whether your software design approach is object-orientated, based on data-flow, or based on conceptual modelling.