



## **THE OBJECT**

**John Ashbaugh**

July 26, 2001

Several years ago, still with the Human Services Research Institute, I was part of a team helping agencies that administer and deliver services to persons with developmental disabilities prepare for the latest movements in the field: managed care, self-determination and performance management. Agencies were particularly concerned with their outmoded or absent information systems, systems needed to manage stepped-up requirements in record keeping, billing and reporting.

A significant portion of the team's time was spent searching for good software. We first looked for functionality and cost. Specifically we looked at affordable software that could perform the prior authorization, individual planning and budgeting, and outcome tracking and reporting functions integral to managed care, self-determination and performance management respectively. However, in the course of analyzing these software options, we came to realize that functionality and price, alone, weren't enough to justify a procurement decision. A used jeep may be the right vehicle for those winter outings in the mountains, but how long would the jeep last, what would it cost to maintain, how would it handle the rigors of city driving and hot summer vacations, and could it hold a growing family. We needed to ask the same questions of the software. Is it maintainable? Will it last? Is it flexible enough to accommodate changing organizational needs. Is it scalable; can it accommodate organizational growth, decline?

We did ask. Nearly all of the affordable software that addressed these front-burner issues was outmoded and/or inflexible and therefore, high maintenance. Outmoded as they were written in archaic languages no longer used by programmers today. Inflexible in the form of hard-coded-procedures and data structures not amenable to change—functions and databases, too inflexible to retain much of their relevance and usefulness to developmental disabilities and other human service agencies over time.

Indeed, we realized that more often than not, agencies seeking our help with the development of their information systems already had systems, most still on the market, but systems that no longer met their needs. This recognition led to another: that our consuming interest in software that could address the hot challenges of the day—managed care, self-determination and performance management—was short-sighted, misplaced. These challenges would sooner or later recede; others would come to the fore. This is not to say that software doesn't have to address present needs, only that it is of tantamount importance that the software be equipped to address whatever other needs arise down the road. Already, we see such shifts taking place. The recently published Transaction Standards governing how Medicaid and other providers will be paid according to the Health Insurance

Portability and Accountability Act (HIPAA) embrace procedure-specific fee payments. This is likely to slow the movement to individualized cross-service budgets and payments favored by the self-determination movement. In many states, the emphasis on cost containment through prior service authorization is taking a backseat to service development efforts fueled by court-ordered or threatened mandates to serve persons awaiting service. Given extreme demand and worker shortages, the heavy emphasis on consumer outcomes is likely to be broadened to include an emphasis on employee recruitment, qualification and retention outcomes.

Our search thus led away from software packages boasting ready-made solutions for today's problems. We were drawn to object-oriented software primarily because of its purported flexibility and easy-to-change nature in the ever-scrambling world of human services.

An object is a software module that contains a collection of related procedures and data. Each object has a basic purpose or function it performs (written as code). Unlike non-object oriented software where code and data are separate, each object contains the data used in performing the function as well as the code. It is self-contained. Users make use of each object through a defined set of messages, and can link the objects to one another to perform larger and broader functions.

The rise of object technology in the building of software products is the software equivalent of the industrial revolution. In the industrial revolution, the move was from handcrafted, individualized products to the manufacture of products from standard interchangeable parts in order to lower the time and cost of production and maintenance. In the software revolution, the move is from handcrafted software to the manufacture of software from standard interchangeable objects thereby lowering the time and cost of production and maintenance.

Constructed properly, software comprised of objects has many advantages over non-object software comprised of routines and subroutines. It is more stable and less prone to crashing when changes are made, particularly in the case of complex systems. Because the code and data are co-located within each object, they don't fall out of synch when the software is changed—as tends to happen with traditional non-object-oriented systems where code and data are separate, and where routines and data tables are highly intertwined.

It is easier to integrate with other software applications that an agency may be using; it is possible to "wrap" many such applications to make them look like just another object controllable through messages. The objects can also be set-up to respond differently to different conditions through the definition of rules. They can dynamically balance network workloads by automatically seeking out idle capacity (computers) on which to execute—a feature that will only grow in importance with internet-based systems. Most importantly, application functionality and databases can be changed much more quickly and at much less expense than non-object software through the elimination and addition of objects, object classes and subclasses.

We find it hard to disagree that the future in software development belongs to objects, however, it is also evident the software revolution is still very much in process. It's only recently that the refinements necessary to deliver on the promise of object technology have been devised. As the market is now insisting on object technology, all of the major vendors of the relational database management systems (RDBMS) prevalent today are investing heavily to extend into object database management systems (ODBMS). There has been substantial investment in an object-oriented version of SQL (OSQL). There has also been the development of Binary Large objects (BLOBS) to incorporate multimedia information. Undoubtedly both object and relational data base management systems will be around for many years to come and in many cases will be employed together—call them ORBMMS with the ODBMS's serving as the front-end of one or more RDBMS's

Finally, we learned that object-oriented software is superior to conventional software only to the extent that it is designed to capitalize on the powerful advantages of objects. The design of object software is in the form of metamodels—representations of the software components, principally software objects—designed to support the operation of an industry and / or agency of concern. If the

metamodel is locked into conventional applications devoted to current problems, the object software loses its hallmark flexibility.

Convergent engineering treats software design and business engineering as one coherent, integrated discipline. Convergent engineering is dedicated to the effective use of object technology through the construction of advanced metamodels. The job of the convergent software engineer is to build a solid set of objects—software “Legos” you might say—where users themselves can configure tailored solutions befitting their needs.” Convergent engineers are trained to look beyond the here and now, designing the software to anticipate the world of situations and changes that an industry or agency is likely to face.

Danic Tools, one of a number of ORDBMS we reviewed, was far superior to the others when it comes to human service applications. The Danic Meta-model is elegantly structured to enable human service agencies to fully capitalize on the power of object technology. Danic engineers spent the better part of a year in close collaboration with human services practitioners just on the basic design of a human services “meta-model” that they felt would serve the full spectrum of agencies well in the present and future. Only then did they set about building the sets of software “objects” necessary to support them.

The central object in case management software is the client or patient; in HR systems, staff; in contract management systems, providers; and in fund raising systems, donors. The central object in Danic is the individual who might have any of the aforementioned roles: client, staff, provider, donor and more (e.g. volunteer, applicant etc.). This singular design means that Danic can be configured to include or integrate with the full range of information system functions needed by agencies in the person-centered human services industry, and effectively eliminate the need to enter information more than once, (e.g. entering the address of an individual who happens to be both a client and staff member into both the case management and human resource systems or modules of the same package).

Danic Tools includes generic objects applicable to the full range of human service agencies: developmental disabilities, behavioral health, child welfare and family services, early intervention, special education, juvenile justice, substance abuse, social service volunteer, and employee assistance programs in Canada and the U.S. It is ideal for the increasing number of agencies providing a variety of services and supports to different populations.

- The **Group object** allows users to sort individuals in the database by any number of criteria for reporting and billing purposes.
- The **File Number object** allows users to encode individuals or groups of individuals and related data as needed for security purposes; it is Danic's key to compliance with HIPAA security requirements and requirements for unique patient, provider and employer identifiers.
- The **Schedule object** allows users to coordinate schedules of any and all individuals in the system as well as vehicles, equipment and facilities.
- The **Facility object** allows users to bill for as well as schedule vehicles, equipment and facilities.
- The **Contact object** allows users to record any and all services and activities and events for management and billing purposes (where applicable) together with related progress notes
- The **Cases object** allows users to organize the contacts into any number of categories for purposes of storage and retrieval, and to define individual problems/ issues, goals and outcomes) related to each of the cases making up the individual's plan.
- The **Contracts object** allows users to set invoice and payment rates, limits and rules for any individual, group, facility, service, case or combination thereof. It accommodates every possible type of payment arrangement and condition.
- The **Questions object** allows users to specify any and all information to be captured in connection with any of the major objects.
- The **Inquiry / Reports object** allows for the retrieval, organization and presentation of every bit of information in the system.

- The **Donor object** allows for the tracking of all donations: monetary, volunteer time and other in-kind contributions, and for the generation of related correspondence, receipt and receivables information.
- The **Human Resources object** allows for the tracking of training, certifications and other employee information and will soon allow for the tracking of time, paid and unpaid, billable and unbillable, as necessary for the management of payroll and benefits.
- The **Production object** (under development) allows for the tracking of client job-shop time and activities, related inventory and costs, as necessary for production management, reporting, invoicing and client payroll.
- The **Billing object** (under development). Agencies receiving Medicaid, Medicare and private insurance will soon be required by HIPAA to meet standard transaction requirements designed to simplify and reduce the time and expense that providers now incur in meeting the widely varying state, local and private billing and related eligibility requirements. Heretofore, Danic has had to configure special billing and report formats for agencies in different states and locales. This standardized object will greatly reduce related Danic setup costs paid by agencies to satisfy their billing requirements.

Danic's meta-model is open to the building of whatever new or modified objects (functions) might be needed over time. I am convinced that object-oriented software such as Danic is destined to grow from the ashes of the popular software of today—software highly structured (hard-coded) to meet the present needs of particular industries in the case of commercial packages and of particular agencies in the case of custom packages. So convinced that I've taken a leave of absence from the Human Services Research Institute to help promote the use of Danic Tools in the United States. I expect Danic Tools to become the software of choice in the human service industry.