White Paper: Object Databases and Jasmine Development Options

V1.00, 7 April 1998.
Peter Fallon & Paul Piko

In the Beginning…

Jasmine is the latest of many products from vendors attempting to create an ***Object Database Management System***. So what separates Jasmine from these other packages that few of us knew, let alone remember with any detail?

The difference is flexibility; not locking the developer into any one development language, environment, or set of requirements. It comes with sufficient visual tools for any data administrator to pick up with ease. The database engine will run on a variety of server systems, including Windows NT and Solaris (Unix), so it's available to the majority of corporate computing environments, and built to handle corporate-level loads. On top of this, you can take home the complete database engine and development tools on a single CD free of charge allowing you to experiment and prove the technology will work for you before committing your budget.

I want to go over some background here, and place Object Databases and Jasmine in the bigger software development picture. As I've already said, Jasmine is an object-oriented database. As such all the benefits of object orientation, that we've become familiar with in VO, are available to you, as integral part of the database. Now you can use encapsulation, inheritance and polymorphism not only in your application programs but right through to the database. In fact your database can now become a far better representation of the real world because it now has the ability to store the attributes, behaviours and relationships of those complex components.

And when you use object oriented techniques to analyse your client's needs and design the data model you no longer have to go through the "artificial" step of translating the information into a relational model.

Object Databases vs Relational Databases

In the early years of the computer industry, programming was largely an art form, requiring a lot of sheer talent. Programs were often monolithic, and the available versions of Fortran, Cobol, PL/I, Assembler and their contemporaries did not encourage much structure in code. A number of initiatives during the 70's brought the programming world around to the idea of **structured programming** and its advantages. Niklaus Wirth's 1976 book: **Data structures + Algorithms = Programs** being a classic example.

Differences between database entity diagramming, and the processes used to create structured programs led to the next advance in software design. We now see that perhaps Wirth's title could be improved to say: **Data structures + Algorithms = Objects**, and that **Objects + Relationships = Programs** (Cossin & Husnian, CA-Technicon Session Proceedings 1993, p415).

Structured analysis concentrates on separating data and functionality. The flow of data and the functions that operate on it are charted and described *separately*. Common structured tools and processes such as dataflow diagrams, entity diagrams, and top down decomposition will be familiar to many of you.

Object Oriented Analysis has a different focus however; it looks to the *combination of data and its associated behaviour*. I think the OO approach is a more holistic one – it tries to look at the business problem as a single entity. Any units we break the problem into should closely represent real world objects. Data and or behaviour related to the same conceptual unit should be kept together because that's how we naturally tend to think of them.

The major ***object oriented*** development languages around today include: C++, CA-Visual Objects, Delphi, Powerbuilder and Smalltalk (this list isn't complete – but they are the main PC products you will come across). Other important languages that are *object based* (they are almost object oriented, but lack certain features) include: Visual Basic, and the toolsets from Oracle and Gupta (I'm sure I've left some out here, so forgive me for not being exhaustive).

If we have an object oriented design for our project/application that is understood and agreed upon by both the business client and the technical team, then there are few, if any, communications problems. This often breaks down as soon as technical design and specification begins – as we try to make that model work using the programming and database tools at hand.

Using any of the above objected oriented programming tools, we can continue using the results of our Object Oriented Design (OOD); usually a set of Class descriptions and relationship diagrams, with assorted notes; and use them directly in our application coding. Classes on paper become classes in our code – and discussions with the client on progress are easy, as we can point to the OOD documentation and indicate directly what has been accomplished to date.

The database design is another question however. And this is where a product like **Jasmine** comes to the fore.

Say you are designing a classic order entry invoice system. I expect that the final object designs, and thence database design will fall into place without too much pain and anguish. This is an example where the data requirements fall straight into a classic relational design without any trouble whatsoever. Too often however, we find ourselves struggling with the database design component of a project. Have you ever had a system where one or more of the following has occurred:

- There has been a high percentage of multimedia data types, or binary data (eg: whole files).
- The final data design contains many small tables, generally related in aspect, and all with structures that are highly similar, but not quite the same. Designing screens and reports for these is tedious because they all share a common basic structure, but there are enough differences to force different screens, code…etc for each.
- The data design has been extremely difficult to arrive at, and there are problems achieving consensus with widely different views on a suitable (normalised) database design among team members.
- The data design is forced to depend on a number of crucial programming "tricks" to work. There are few areas where a standard (individual) SQL query is sufficient to extract the data required without programming intervention.
- Sheer complexity. Relational databases are difficult to handle and document as they get more and more complex. If the original problem is extremely complex, simple data relationships don't solve the business problem, and program code becomes the only way many tables are linked, filled or used. OO techniques help with the program side, but the database remains a problem.
- Any combination of the above, which results in a project only being feasible or achievable if you get you top, top developers on it, who are familiar with bleeding-edge technology. Even if you proceed, there are expected problems with maintenance and ongoing enhancements requiring the same top-notch staff.

I suggest that these are classic instances where you were perhaps forcing a round *Object* peg into a square *Relational* hole. The business objectives were probably straightforward; complex perhaps, but they could be described in english with no communication problems. There may even have been an original, elegant manual or paper based system that was replaced, but the final computer system was, at the very least, awkward, and never achieved the elegance or simple functionality of the old method!?

You may find it a useful exercise to return to one of those old systems, revisit the original analysis documents, and see what type of Object Oriented Analysis/Object Oriented Design you can come up with. You may be surprised.

What I am getting at here is that when a project gets to the technical design and specification phase, we tend to fall back to *structured* techniques. We break off into two camps: the programming camp, which looks at the code, screens and reports; and the database camp, which constructs the database designs, generally using relational analysis, or *normalisation*.

Classic systems such as Order Entry/Invoice, or others with a high number crunching content (to use a stereotype) may fall into the relational data model with ease. In fact the object/class design will have obvious similarities with the relational database's table structures. Many times however, the problems listed above will occur, and relational data analysis is a struggle, or the final designs are unsatisfactory, inelegant, or forced. The cost is that a lot of code will be written to handle the interface between application objects and database storage. This resulted in increased development time, increased testing due to complexity, increased project costs, and a larger lifetime cost, as any subsequent changes will take longer to implement and test.

At these times we should look to an Object Database such as **Jasmine**. The reason – an Object Database's "database design" is essentially the results of our OOA/OOD process! Using a pure Object Database can reduce this complexity, and leave you time to deal with the complexity of the business problem at hand, which is what you should be focussed on.

Please remember one very important point. Object databases do not supersede Relational databases. Each are designed to perform certain tasks very well (and the grey areas around the edges are mostly due to the experience of your programmers!). Use the above rules-of-thumb as a guide to which may be the better solution in any given scenario. At the same time, don't confuse complexity due to scale/size with the difficulties I listed – a large system is going to be hard to implement no matter what the database!

Jasmine, an Object Oriented Database!

Using an Object Oriented Database (OODB), the results of Object Oriented Design such as classes, instances and methods are directly utilised. Classes become database *Classes*; Class relationships, especially "is kind of" relations allow us to define *Class hierarchies* and an inheritance structure in our database. Properties on paper become Instance properties, Methods become ODQL and API methods in the database.

The real decisions we are left with revolve around three issues:

1. Which objects/classes in our design have to be ***persistent***. That is, which must be in the database, and which are only required at runtime within our program code? If its data you previously stored in an SQL or DBF table, then its going to become a persistent class on the Jasmine Server. Any object only required at runtime can be safely defined solely within our application.

2. Which methods will be coded into the database classes (using ODQL or API calls), and which will be coded into the matching classes in our program code? This is a basic question of ***partitioning***: What parts of the system should reside on the server, and which parts should reside on the client!? Anything you previously coded as stored procedures, database triggers (inside VO or on the database) are prime candidates for methods. Any data-intensive activity should be done in a method so it executes on the server, and does not drag huge amounts of data across your LAN.
3. Should we be using ODQL or API-level language code for our methods? If you require API programming you need to look for staff with C/C++ skills.

If we have problems other than these, we need to re-evaluate our analysis and design process. Just as a programming problem may highlight poor or inadequate designs, problems integrating the design into the object database may indicate similar issues.

Remember that there is a distinct difference between a poor design, and one that is complex simply because the business model is complex. The business/clients should be with you every step of the way now, as you are now using a modelling process they can easily follow with little explanation (you should still be using *their* terminology to describe these classes and objects!). If you have to *iterate* back to an earlier phase to sort out an issue, then they must be involved just as they were the first time.

Handover to maintenance and support staff will now be much easier, as object-oriented designs constructed early in the project are now consistent with the entire system, programs and data alike. Reading, and understanding the original business requirements, and then going onto the technical designs should be a much smoother transition for staff unfamiliar with the project. If you have used OO techniques well, you should even see a reduction in maintenance costs over time, as making changes to code or data will be a simpler process.

The ***paradigm shift*** from relational design to object oriented design for databases will take a retraining on the part of your staff, and some time for them to see the advantages, and just how the newer schema works. I call it a *paradigm shift*, as it requires technical staff to think of data storage in different terms – it's a change in the way they approach a problem and look for a solution. If your team is already experienced in OO techniques for coding, then you have an advantage.

This is identical to the shift required of programmers several years back when most mainstream programming languages (such as Clipper) began offering OO extensions, and they had to change from the older *structured programming* techniques to object oriented programming. There were, and still are, cries of "*I don't understand!*", as each of us waited until the penny dropped, and we exclaimed "*ohhh, but that's so easy, that can't be right!*".

The following table is a collection of analogies which are NOT perfect, and any purist will have problems with them. However, they will assist you to wrap your mind around where some of the concepts fit, compared to how you may work now with relational technology:

Object Database Components	**Relational Equivalent**	**Discussion**
Class	Table	
Class Family	*None*	There is no relational analogy for a Class Family, as it is specifically a Jasmine concept (based on Class Heirarchy). One analogy is if you keep separate Test and Production copies of your database and configuration or INI files, or separate business data and program data into different databases. These could be separate Class Families in Jasmine.
Object/Instance	Record/Row	
Instance Property	Field/Column	
Class Property	*None*	Class Properties are generic, class level values that are not part of any specific instance, common to the entire class. There is no relational equivalent, excepting deliberate constructs of data and stored procedures in separate reference tables.
Instance Method	Stored Procedure	Both of these are pieces of code which can affect the data stored in the database. Here the similarity end however, as Methods can affect information outside the database, but only only for a particular object/instance. Stored procedures can affect any data, but are limited to acting on the database itself.
Class Method	Stored Procedure	Like Class Properties, Class methods operate at a generic level – ie: they work on the Class rather than specific instances of it. Stored procedures can have a similar scope though this will vary from product to product.
ODQL	SQL	ODQL is far more powerful an language, and methods compile to C DLLs, so its very fast as well.
API language in methods	Program code in client applications.	

Jasmine Development Options

An important feature of Jasmine is that, in addition to providing these object oriented benefits, there are many and varied ways for you to access and manipulate the objects stored in a Jasmine database. This provides you with many choices on how best to meet your client's needs and gives to flexibility to solve a wide variety of problems. For this reason, "*Where do I Start?*" is a common question, and a sensible one.

Do you want to produce a flashy multimedia application that can be run stand-alone or as a web browser plug-in? Then you'd use the tool what was called JADE on the developer CD (which is now called Jasmine Studio). Do you want to access the database in HTML pages through CGI/NSAPI? Then WebLink will do the job. Do you want to view and update the data from a Windows application? Then the ActiveX control or the API approach might be the way to go, with the choice of a number of languages/tools. And there's more...

The thing is, you have all these ways that you can create the solution you are trying to provide. Maybe you'll use one of these tools, but more likely you'll use a combination of them, as best suits your needs.

Of course, whichever way you choose to write your applications, you still need to follow OO techniques to come up with an appropriate database structure as I described earlier. You then implement your OO model using a combination of the Jasmine Database Administrator and ODQL (the database's programming language). You create and build your classes, define properties and write methods (in C/C++/ODQL).

The following table describes the key technologies in Jasmine, and the areas of modern application development they are key parts of. I mention the word *Key* deliberately. eg: n-Tiered development will incorporate a number of elements, including any of the programming options (Jasmine Studio, ActiveX, JAVA or API interfaces), but the key to *separating business logic from the interface* is ODQL, and programming methods to execute on the server.

Technology Requirement	**Jasmine Options**
Kiosk-style application, quick prototyping	Jasmine Studio
Client Server Development	Jasmine Studio
	Jasmine C/C++ API
	Jasmine ActiveX Control
n-Tiered Database Development – Business Logic separated from Applications	Jasmine ODQL (Object Data Query Language)
	Programming Jasmine Methods using API and embedded ODQL.
Object Database Environment	Jasmine Database
Ability to handle multimedia and complex data types	Jasmine, Multimedia Class libraries ship with product
Integration with legacy SQL databases	Jasmine, SQL Class libraries ship with product
Internet/Intranet development	Jasmine Studio (with Netscape or Internet Explorer)
	Jasmine ActiveX Control & Internet Explorer
	Weblink
	JAVA links to Jasmine (3 options from various parties)

Jasmine Studio

If you have the developer's CD, or even a fully copy of Jasmine, you are probably wonder what this is. Well, it's the new name for the development tool previously called JADE. This is the point-and-click, no-programming-required development environment that CA provides with Jasmine.

To me, Jasmine Studio's prime purpose is for developing kiosk-style applications for information dissemination, or operating electronic commerce applications. It can create traditional EXE files for distribution on a LAN, or internet enabled application for intranet or internet use. Its power in this area cannot be underestimated; its very strength is the fact that all business logic resides on the Jasmine server in methods – it cannot perform anything other than interface operations and data display.

If you can completely separate the business logic from the interface of a project, and the interface is not complex, then Jasmine Studio may be ideal for you. Certainly if you want to do any internet or intranet stuff with Jasmine, its your first stop.

I admit, the internet is an area I am weak on at present (and I dislike writing about a subject I have not fully worked through and created something with!). I will be attending a number of sessions on this at CA-World. See you there!

JAVA

Finding information about the JAVA interfaces is rather difficult. The best information I have is from a session I attended at CA-Expo in Melbourne last year. The Fujitsu speaker spoke of three Java interfaces to Jasmine, all of which

were produced by independent vendors: The first is known as the *Persistent Java* interface which consists of a runtime library that is distributed with your Java application. It offers transparent object persistent in Jasmine.

The second is known as *Java Proxies*. This offer tighter integration with the Jasmine engine, database operations such as transactions, and the data types supported in the Jasmine database than the Persistent interface.

The third interface is *Jasmine Java Beans*. A set of Java Beans that encapsulate Jasmine functionality have been created to make development easier, and less messy than the first two options. It also integrates well with the latest tools and methods used in Java development.

There was also a JDBC interface spoken of in the Jasmine launch in Australia. But I have not been able to find any written materials to state whether that is part of the above, or a separate interface again.

If you are interested in this, look at the CA web page and speak to your local rep to find out further information. When I get some more myself, I'll update this paper.

ActiveX Development

ActiveX development for Jasmine is used for two goals: Application development in a traditional client-server sense, and Internet/Intranet programming through Internet Explorer. My emphasis here is on business application development.

The Jasmine ActiveX control provides an easy-to-use interface to the Jasmine database, avoiding all of the complexities of the C API, ODQL programming, and the like. Of course this comes at a cost; there are many Jasmine features not available to us directly. You can still use most of this advanced functionality, but it requires a firm understanding of Jasmine, Object Oriented Design, and ODQL to construct the workarounds.

That said, you can still build quite complex applications using just the ActiveX control. It remains a serious option for any development, simply because your most involved tasks are normally encoded into methods.

Any ActiveX compliant language/tool can use the Jasmine ActiveX control. Just think about how many tools and products we are talking about here!

API Level Programming

If your development language supports standard Windows DLLs, and the data types necessary to call C functions (ie: Pointers), then the API is a real option. Using the API you have the entire Jasmine engine at your disposal, including a runtime ODQL interpreter (allowing ODQL statements to be constructed and executed by your program outside of discrete methods). This goes way beyond what is possible using the Query construct present in the ActiveX control.

As you'd expect, there are definite speed advantages using the API as well as the sheer programming power. The cost is equally obvious – complexity of programming. You have to pay far more attention to detail, memory allocation and other factors when using the API. Whether you will opt for this approach, or to program all/most application complexity using methods instead is a decision you will make based on Analysis/Design requirements and available staff skills. Note that the latter approach is essential if you wish to use the ActiveX, or the Jasmine Studio for the client end application.

CA-Visual Objects Options

CA-Visual Objects is Computer Associate's flagship PC development language. Many of you may remember Clipper, which was the leading xBase development language in the DOS world. Well, think of Visual Objects as Clipper's successor in Windows (its not the same, though the language has many similarities). Visual Objects combines the flexibility and high level design abilities of say VB and Dephi with the low level power, true object oriented programming and native compiler speed of C++. As such, it integrates with Jasmine rather well, and supports both an ActiveX and API approach to development. And with the work that CA is doing in integrating VO/Jasmine, in addition to some still to be released third party products, the distinction between a VO object and a Jasmine object will become even less.

Piko Computing Consultants Pty Ltd has a small VO-Jasmine sample up on their web page **http://ourworld.compuserve.com/homepages/piko** If you want a start on using Jasmine from VO see the course details below.

Visual Basic Options

If you are using Visual Basic for application development, you will have designed and constructed your Jasmine database, and placed most of the complex logic in Methods. If specific or complicated data retrieval combinations are required, methods will be constructed for those as well in the same way you would construct stored procedures in a

relational database. Visual Basic simply provides the front end to the database, and the data handling and validation necessary to accept data entry, and provide feedback on user requests.

The above is a very cut-and-dried, and oversimplified statement. In most modern applications, the user interface is a very complex construct. The advantage here is that a properly designed object database allows you to use Visual Basic where it is most effective – flashy screen design and user interaction. The ActiveX interface, which is the only practical one to use in VB, forces you to keep database interaction straightforward, and keep you focussed firmly on the client side of the n-tiered application.

If you want a start on using Jasmine from VB see the course details below.

Jasmine Courses and Tools:

CA Development Partners Castle Software Australia and Piko Computing Consultants are pleased to announce the following course notes are now available:

- **Jasmine ActiveX Development with CA-Visual Objects**
- **Jasmine ActiveX Development with Visual Basic**

These are the first installments in a series of courses covering Application Design and Development with **Jasmine**, the industrial strength, object-oriented database from **Computer Associates**.

These notes combine white-paper style lecture notes with exercises designed to introduce you gradually to the ActiveX interface to the Jasmine database engine. Work at your own pace to build an application that covers all of the major functions you are likely to need in your own projects. Instead of puzzling out how something works by looking at the documentation, watch it happen in your own program, and understand Jasmine faster!

Fully worked solutions (source code) are provided for all exercises.

Please contact us for Pricing and ordering details. Our details are below.

Further Courses

Drawing on their extensive experience in solving real-world business problems, **Castle Software Australia** and **Piko Computing Consultants** will be providing courses and products designed to match the needs of application developers.

Here are some of the courses that will be available soon:

- Object Oriented Analysis and Design
- Installing and Configuring Jasmine Databases
- Jasmine Database creation and ODQL Programming
- Jasmine API development with Visual Objects

Announcing Jasmine API class library for CA-Visual Objects!!

At CA-World, we will be demonstrating a Jasmine API Class library that will provide API speed, functionality and flexibility to CA-Visual Objects. It will include: creating VO objects that match Jasmine data objects (make your own jServer{ } now!), ODQL programming within VO, without having to worry about C datatypes, pointers or structures. We'll also provide all of the necessary prototypes if you want to get down and dirty as well!

There's much, more, but we want to surprise you! Talk to us at CA-World 98 and ask for a demonstration of this product, or to peek at our course materials!

Authors:

If you've got any more questions, or would like to know about Jasmine training, course notes and tools (Jasmine and CA-Visual Objects) please contact either of us. We are:…

Paul Piko
Piko Computing Consultants
8 Hedline Place, Macleod, VIC, 3085
Australia
Ph: (+61 3) 9432 1222
Fax: (+61 3) 9432 1255
piko@compuserve.com

Peter Fallon
Castle Software Australia
3/8 Reid St Ashwood, VIC, 3147
Australia
Ph: (+61 3) 9885 5184
Fax (+61 3) 9886 0100
pfallon@compuserve.com

Copyright Notice

Trademark Acknowledgements