ODBMS.ORG User Report No. 24/08

Editor Roberto V. Zicari- ODBMS.ORG www.odbms.org
October 2008.

Category: **Industry**
Domain: Various
User Name: **Thomas Amberg**
Title: *Software Engineer*
Organization: **Oberon microsystems, Switzerland.**

Thomas Amberg, graduated in Computer Science at ETH Zürich in 2001, working as a Software Engineer at Oberon microsystems since then.

Q1. Please explain briefly what are your application domains and your role in the enterprise.

Thomas Amberg: Oberon Microsystems (www.oberon.ch) is a small company providing mobile and embedded software engineering, systems and software architecture to R&D departments of corporate clients. Applications range from power plant monitoring to mobile data collection to wireless body area networks. Modularity (David Parnas), simplicity (Niklaus Wirth) and Design by Contract (Bertrand Meyer) guide us to deliver innovative yet highly reliable and robust solutions to demanding engineering problems.
Our software is based on C# and the .NET Compact Framework and .NET Micro Framework platforms.
As a Software Engineer my work ranges from software architecture (e.g. connecting mobile devices to a server on the internet) to design tasks (e.g. specification of protocols and formats for data exchange) to implementing and testing a prototype or first version of a product.

Q2. When the data models used to persistently store data (whether file systems or database management systems) and the data models used to write programs against the data (C++, Smalltalk, Visual Basic, Java, C#) are different, this is referred to as the "impedance mismatch" problem. Do you have an "impedance mismatch" problem?

Thomas Amberg: Mobile and embedded systems with slow battery powered hardware require a fast and reliable persistence mechanism. Due to memory constraints, each application must expect it's process to be killed without prior notice. Immediate, incremental persistence of changes is therefore of utmost importance. The amount of data to be stored is often rather small and only a limited number of simple queries are required. To meet such requirements we use our own object persistence solution.

For data transmitted over system boundaries to be stored on a server, an "impedance mismatch" between the internal object representation and the data "on the wire" exists, but does not pose a problem. We prefer language-independent, open and accessible representations (e.g. in XML or JSON) over binary object serialization formats. To convert data to and from internal objects, a simple sequential Reader/Writer is less resource intensive than e.g. a DOM parser.

Hiding system boundaries behind an internal data model (e.g. using RPC or .NET Remoting) is often not a good idea, because of the much higher latency and unavoidable communication errors that have to be addressed by client code anyway.

Q3. What solution(s) do you use for storing and managing persistence objects?

Thomas Amberg: We use a custom object persistence solution based on sequential serialized update operations appended to a binary file. The idea behind this design is described by D. Birrel et al. in (birrell.org/andrew/papers/024-DatabasesPaper.pdf). Client code accesses the persistent objects through regular C# interfaces. The implementation of those interfaces is generated at compile time. The code generator consumes C# interfaces annotated with implementation hints (custom .NET Attributes) and produces C# source code.

What experience do you have in using the various options available for persistence for new projects? What are the lessons learned in using such solution(s)?

Thomas Amberg: The above approach limits the size of an object graph (as the whole graph remains in memory) but is extremely fast when it comes to storing changes. Often it is not even necessary to "compact" the stored changes. Care has to be taken when "unloading" an object graph to prevent "dangling references". Finally, while using a code generator imposes certain conventions concerning the object model, it speeds up implementation and reduces testing and debugging time.

Q4. Do you believe that Object Database systems are a suitable solution to the "object persistence" problem? If yes why? If not, why?

Thomas Amberg: Yes, if the requirements allow for it. Not having to use database products can be an advantage as it reduces overall complexity of the system and in some cases saves license costs.

Q5. What would you wish as new research/development in the Area of Object Persistence in the next 12-24 months?

Thomas Amberg*:* The main problems arising in our case are replication, memory footprint and to a lesser degree queries. There are some good papers on replication, and for queries (at least in C#) LINQ looks promising.