InDetail

Bloor

# expressor 3.5 to 4.0

An InDetail Paper by Bloor Research
Author : Philip Howard
Publish date : December 2011

If ease of use is a major consideration for you (and it should be) then expressor is worth serious consideration

Philip Howard

## Executive summary

Talk to any vendor in the data integration market and they will tell you that their biggest competitor is hand coding. Why? The simple answer is that many users do not believe that the capital costs involved in investing in data integration can be recouped through the reduced development requirements and performance benefits that can be derived from using a tools-based approach. expressor has set out to challenge this assumption through a semantically-aware product that makes development much simpler. Put simply, an understanding of semantics enables you to automate data integration and ETL (extract, transform and load) processes that cannot otherwise be easily automated and, of course, this also makes reuse much easier. This, in turn, makes development simpler which, together with an attractive pricing model and a focus on high performance, is expressor's answer to those that were previously committed to hand coding or, for that matter, competitive products.

### Fast facts

The key differentiator that marks expressor out as different from the rest of the data integration crowd is its understanding of semantics and it is important to understand how this works. In fact, expressor works on the basis of semantic types. This does not require any significant work on the part of the user and it means that the development of data integration processes is much simpler and it also makes reuse easier.

The company has three editions of its product currently available: the Community Edition, which is available for free download; the Desktop Edition which is a "try before you buy" version, and the Standard Edition. A fourth product, the Parallel Edition, is expected to be available from February 2012.

### Key findings

In the opinion of Bloor Research, the following represent the key facts of which prospective users should be aware:

- Semantics is at the heart of expressor. In particular, it enables both faster development and improved reuse by separating semantic type definitions from the dataflows that they are used by.

- A traditional graphical approach to development is provided. A large number of pre-built expressions are included with the product (you just have to complete relevant parameters) and there are also facilities to develop your own libraries of such functions.

- Where required you can use expressor Datascript. This is an extended subset of the open source Lua language. For companies not having Lua skills this will require some education though Lua has similar semantics (though not syntax) to JavaScript. Lua does have the advantage that it can also be used at the operating system level.

- There is an in-built rules editor that makes it easy for non-experts to develop simple rules. There are a number of ways in which these can be shared for reuse purposes although there is currently no facility to automatically add transformation rules to a Datascript module. This is planned for a forthcoming point release.

- For development and testing purposes, logic is separated from data. In other words, you don't have to know about data structures in order to develop your processes.

- Optimal use of parallelism has been designed as a fundamental feature of expressor but the different editions have different levels of parallelism implemented. The Community Edition has features such as pipeline parallelism but not cross-core parallelism, which is in both the Desktop and Standard Editions. The Parallel Edition will also support partition parallelism and have support for parallel loading.

- The company has introduced an Extension Framework that enables developers to create packaged integration solutions. The company is also introducing a number of such solutions.

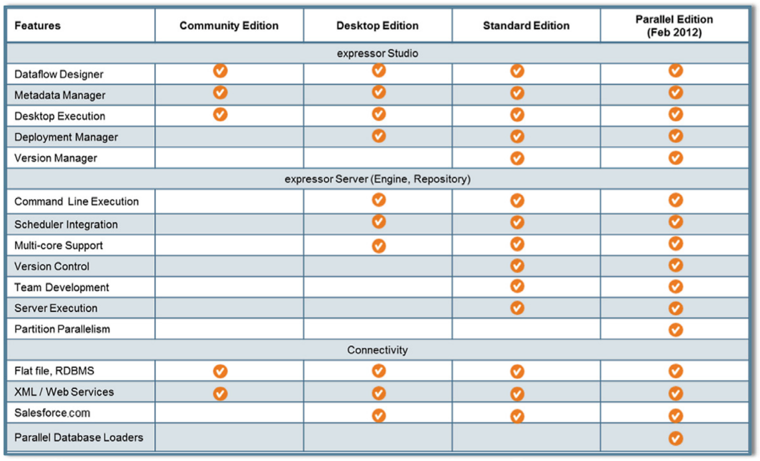
## Executive summary

### The bottom line

When expressor first came to market in 2008 it was targeting very high performance for complex transformations. However, it subsequently changed its go-to-market model to the structure seen today. In our view this was a sensible change. We are firmly of the belief that the use of semantics is the way forward in many technology areas, of which data integration is just one. When combined with the product's extensive support for parallelism it is our opinion that expressor offers significant advantages over traditional methods in terms of ease of use, reuse and performance.

## The product

At the time of writing there are three current versions of expressor: the Community Edition, the Desktop Edition and the Standard Edition, all of which are currently in version 3.5 (as of December 2011). Version 3.6, scheduled for February 2012, will see the introduction of the Parallel Edition. Version 4.0, which we will not otherwise discuss here, will include hierarchical support for XML data. This release is scheduled for the second quarter of 2012. The following table highlights the differences between the different products, as well as showing the major elements that make up the product:



| Features | Community Edition | Desktop Edition | Standard Edition | Parallel Edition (Feb 2012) |
|---|---|---|---|---|
| expressor Studio | | | | |
| Dataflow Designer | ✓ | ✓ | ✓ | ✓ |
| Metadata Manager | ✓ | ✓ | ✓ | ✓ |
| Desktop Execution | ✓ | ✓ | ✓ | ✓ |
| Deployment Manager | | ✓ | ✓ | ✓ |
| Version Manager | | | ✓ | ✓ |
| expressor Server (Engine, Repository) | | | | |
| Command Line Execution | | ✓ | ✓ | ✓ |
| Scheduler Integration | | ✓ | ✓ | ✓ |
| Multi-core Support | | ✓ | ✓ | ✓ |
| Version Control | | | ✓ | ✓ |
| Team Development | | | ✓ | ✓ |
| Server Execution | | | ✓ | ✓ |
| Partition Parallelism | | | | ✓ |
| Connectivity | | | | |
| Flat file, RDBMS | ✓ | ✓ | ✓ | ✓ |
| XML / Web Services | ✓ | ✓ | ✓ | ✓ |
| Salesforce.com | | ✓ | ✓ | ✓ |
| Parallel Database Loaders | | | | ✓ |

All products are currently Windows-based but Linux support will be introduced with version 3.6.

In terms of connectivity, expressor supports native database access to Oracle, DB2, Informix, SQL Server, Sybase, Teradata, PostgreSQL, MySQL and Netezza; generic ODBC connectivity for other environments; XML, flat files, other complex/hierarchical data structures, and ftp. There is also SOA support. Change data capture is supported through a partnership with Attunity. Note that Teradata provides specific drivers for loading data into its warehouse products and expressor offers integrated support for these drivers.

The product consists of three major elements: expressor Studio, the Repository and the Data Processing Engine. We will also discuss the Extension Framework, which is a major feature of the latest release. However, before discussing each of these (which we will do in turn) it will be useful to consider expressor's approach to semantics, which lies at the heart of the product, and which is executed by the engine.

### Semantics in expressor

expressor is based around the use of semantic types. This provides developers with a graphical means by which to model their data structures, which are known as semantic types. They can be extended to include user-programmed rules for conversion, interpretation, error handling, and so forth. Once applied to a dataflow, a semantic type automatically mediates the variations that occur among various data stores' versions or interpretations of the same data structure. This is best illustrated by example.

## The product

Consider a database for a freight service provider that stores US truck and nautical route distances in imperial units of measure, such as feet or miles. The data in this database needs to be ported to a data mart in France, from which reports will be generated for Indian consumers who expect units to be reported using the metric system. At some point in the future, the freight service provider expects to have a separate database for international routes and from which reports will be generated for the same French consumers. The new, international database will only contain metric units of measure.

To accommodate all the interpretations of 'distance', an expressor developer would develop a model (semantic type) for 'distance' that included all of its various characteristics, such as its standardised datatype and its valid range of values. Additionally, a developer could extend the type to include functions for conversion ('how do I convert feet to metres?'), interpretation ('what if the distance is reported using a scale factor?'), and/or error handling ('what if someone enters a volume measure, such as 'gallon', where a distance measure was expected?').

The important thing here is that the semantic type for 'distance' is defined independently of any processes (dataflows) that might make use of 'distance'. Thus it can be defined once and used in any dataflow that references distances. Once inside a dataflow, the 'distance' semantic type would mediate between all the different types of distances (feet, furlongs, metres and so on) it encounters without asking the developer to hardcode the mediation logic into the dataflows. Later on, if a new unit of distance, such as a (Gunter's) link (7.92 inches or 0.01 chains) is introduced into the data that the set of distance-processing dataflows are consuming or producing, then the developer needs only update the 'distance' semantic type to accommodate the new unit of measure. As a result of updating the semantic type, all of the dataflows that reference it will also be updated as well and can automatically start to accommodate the new unit of measure.

More generally, a semantic type may be defined for an object or entity of interest to the business. Not only measurements but customers, suppliers, parts, products and so on, are all amenable to this approach. For example, suppose that you define a semantic type for 'Account_number' whereby this is defined as a 9 digit numeric field. The way that the software works is that if you map, for example, acc_num in a source record to Account_no in a target record then the software will map acc_num to the semantic type Account_number, performing any data type formatting conversion automatically and then it will map Account_number to Account_no, again automatically taking care of any differences in formatting. The difference between this approach and that of traditional tools is illustrated in figure 1.

## The product

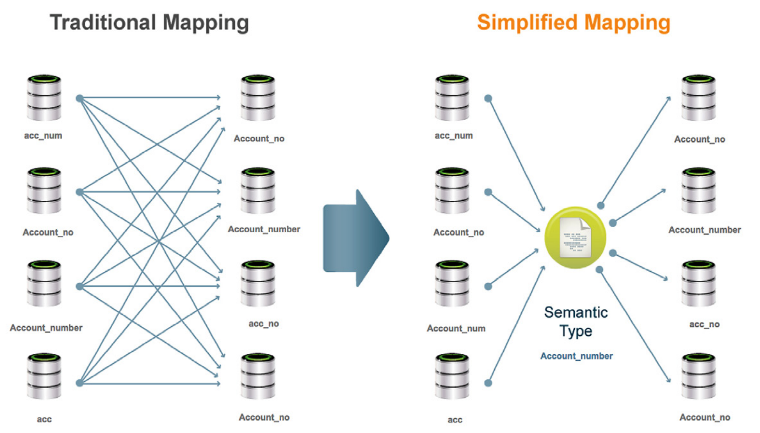
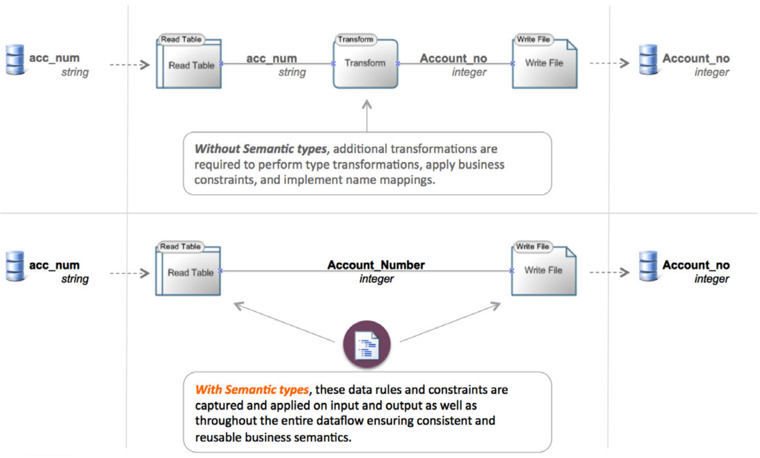


**Figure 1:** Mapping with semantic types versus traditional methods

There are three big advantages involved with this approach. The first is that you don't have to worry about formatting once the semantic type is defined (which you can do iteratively). Secondly, it makes process flows much easier to design and easier to visualise, because you do not have to format transformations as a part of your flows. This is illustrated in Figure 2. Thirdly, because semantic types are independent of dataflows, each is much easier to maintain and each is easier to reuse.



*Without Semantic types*, additional transformations are required to perform type transformations, apply business constraints, and implement name mappings.

*With Semantic types*, these data rules and constraints are captured and applied on input and output as well as throughout the entire dataflow ensuring consistent and reusable business semantics.

**Figure 2:** The use of semantic types simplifies dataflows

## The product

### expressor Studio

expressor Studio is the company's design, management and administration tool. It has the look and feel of Microsoft Office 2010. Historically the product required the use of Microsoft Visio but this has now been replaced by expressor's own diagramming interface.

The logical progression through which you develop processes within expressor is that you start by creating a workspace, within which you create your project. Next, you choose the connection to the database you want to access and then select the elements that you are interested in from the table view that the connector presents you with. The software will automatically recognise any relevant semantic types and will generate appropriate mappings as a part of the input schema. The next stage will be to create the actual dataflow (see Figure 3), which you do in a conventional manner by dragging and dropping relevant operator shapes onto your dataflow canvas. Associated with each operator you can define properties (for read and write operators) or expressions (for transformations) and the penultimate step is to define an output schema. Finally, you create a deployment artefact which includes everything you might need at run-time.

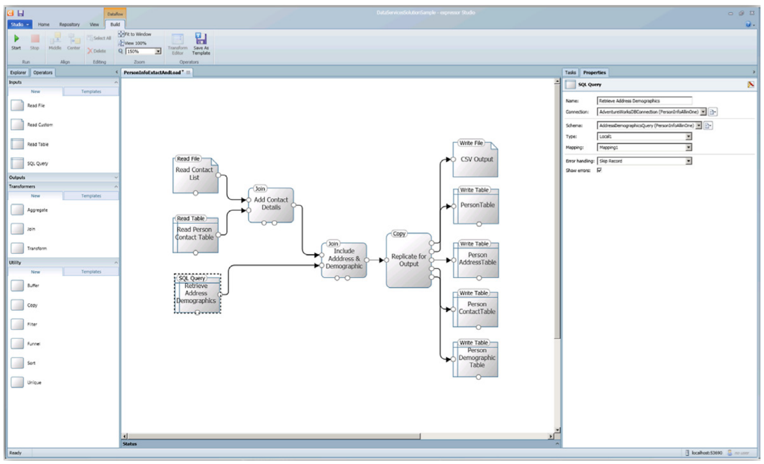


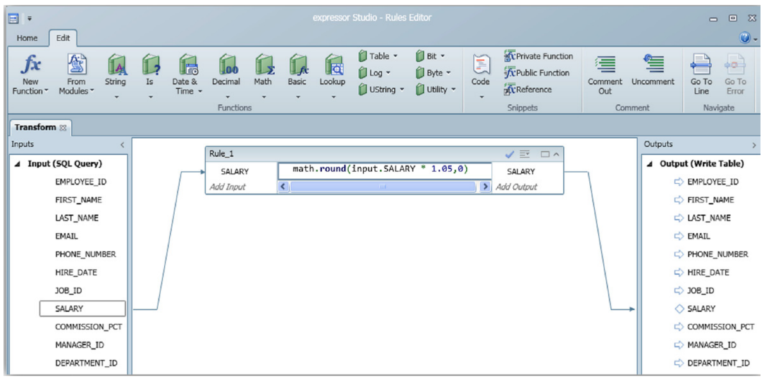
**Figure 3:** expressor Studio

A full set of operators to perform functions such as joining, transforming, sorting, collating (which allows the automated generation of relevant statistics such as standard deviations) and partitioning are provided, though you can also drop into Datascript (see below) for more complex functions if necessary. The resulting scripts can either be embedded within a dataflow operator or written to an external file that is called from an appropriate operator.

Studio offers a wizard-based interface that abstracts much of its scripting language so that this should be amenable to non-expert use, at least for processes that are not too complex. It has done this by providing pre-built expressions for standard transformations where all you need to do is to fill in the appropriate parameters. There are also facilities for users to develop their own libraries of functions to further enable this ease of use. In addition, the expressor Rules Editor (see Figure 4) allows you to define simple transformation rules without having to do any scripting. These rules can then be designated as templates and copied into other workflows. Note that the symbols associated with the target are on the right hand side of the mapping. Elements with an arrow are

## The product

automatically propagated using semantic typing while those to which rules apply are shown with a diamond.



```
math.round(input.SALARY * 1.05,0)
```

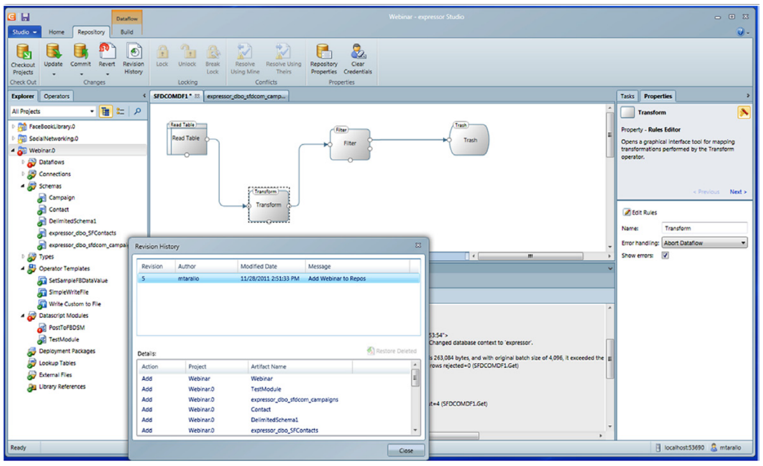
**Figure 4:** expressor Rules Editor

For those that do not wish to develop rules using the rules editor, or for complex rules for which this is not suitable, then you can use expressor Datascript where Datascript is an extended subset of the Lua open source scripting language (which has semantic similarities to JavaScript though rather different syntax).

Studio offers debugging capabilities and also includes a pre-processor that automatically analyses and graphically tells you (visually) if an operator in a dataflow is completely configured or not. Additionally, Studio allows you to test dataflows on a remote system for development purposes.

### expressor Repository

expressor's Repository is tightly integrated with the expressor design environment. It is used to store, manage and support the reuse of all the project artefacts that we have already discussed, including semantic types, business rules, run-time details and so forth. The repository includes the normal sort of version control, branching, snapshots, check-in/out, and multi-developer support (at least in the Standard Edition, version control is not included in the Community or Desktop Editions) one might expect.



| Revision | Author | Modified Date | Message |
|---|---|---|---|
| 5 | mtarallo | 11/28/2011 2:51:33 PM | Add Webinar to Repos |

Details:

| Action | Project | Artifact Name |
|---|---|---|
| Add | Webinar | Webinar |
| Add | Webinar.0 | TestModule |
| Add | Webinar.0 | expressor_dbo_sfdcom_campaigns |
| Add | Webinar.0 | Contact |
| Add | Webinar.0 | DelimitedSchema1 |
| Add | Webinar.0 | expressor_dbo_SFContacts |

**Figure 5:** Users can check-in/out all project artefacts from the expressor Repository, which is tightly integrated with the Studio UI.

## The product

The repository also plays a key role in managing deployment artefacts in production environments. You can create deployment artefacts which contain all of the dataflows that represent an ETL job and check them into the expressor Repository. The repository can be accessed via the command line by your favourite scheduling tool to check out the latest changes to a deployment artefact before execution.

### The expressor Engine

expressor was designed from its inception to support parallel processing and it originally included support for both implicit parallelism (whereby the engine automatically takes advantage of available parallelism) and explicit parallelism (where it is under the developer's control). However, the latter is not, and will not, be available within the Community Edition.

Pipeline parallelism, whereby different (but often related) tasks are carried out in parallel is also supported both for moving the data and processing (depth parallelism) it. However, perhaps most significant is the product's support for multi-core parallel partitioning. This is not available in the Community Edition. Partition parallelism, whereby you can parallelise processing across multiple servers as well as within a server, will only be available in the Parallel Edition.

It is perhaps worth expanding on multi-core or cross-core parallelism. In many traditional environments parallelism is based on multiple processing nodes whereby you have one parallel channel per node. However, in the case of expressor, parallelism is an intra-node capability rather than an inter-node one (except in the Parallel Edition, which does both). Thus the key issue is the number of cores not the number of nodes. The big advantages of this approach are that it is less expensive and requires a smaller footprint in hardware terms and it provides great performance: in its own tests, and using the compression techniques it can employ, expressor has demonstrated load speeds in excess of 10Tb per hour, which is outstanding. It is also worth commenting that the techniques used to achieve this level of performance have been patented so it may be some (long) time before competitors can catch up with expressor in this respect.

Apart from its performance the processor provides the sorts of capabilities one might expect, such as the ability to run in either intermittent or continuous mode (albeit it is not quite as sophisticated as some of the high-end products on the market in this respect) and support for functions like exception handling.

### The Extension Framework

The expressor Extension Framework is a new feature introduced with version 3.5. The idea is to provide an environment to support specific integrations with particular products or environments. So, for example, in version 3.5 there is the release of such functionality with Salesforce.com. This includes pre-built operators, semantic types, connectors and schemas, so that you can link Salesforce.com with CSV files and ODBC compliant databases, either as a source or target.

Another packaged extension that expressor will be supporting in version 3.5 is with Melissa Data, which provides contact-focused data cleansing capabilities so that things like name and address matching can be built into expressor workflows. The company plans to release two further packaged extensions in a point release in January 2012 to support QlikView (QVX) and native Microsoft Excel environments. In the former case, the company claims that the integration will be particularly close. expressor also provides facilities for developers to create their own extensions and intends to encourage users to share such extensions through its Community web site.

## The vendor

expressor was established in 2007 (though the ideas behind it originated in 2003) and came to market with the first beta version of its eponymous product in March 2008 followed by a general release in June of that year. The company is backed by venture capital and completed a 'C' round of funding during 2010.

expressor software has a small field sales team in the United States. More generally, the company's go-to-market strategy is to leverage partnerships, systems integrators (SIs) and resellers. In the latter category, expressor has partners in the UK, Switzerland (covering Austria and Germany as well), Slovenia (covering the Balkans), South Korea, South Africa, Australia and in the United States. In terms of systems integrators, expressor partners with more than two dozen SIs including Sogeti, Novedia, and Acxius. One of expressor's early partners, Bitwise, has developed a migration tool from Ab Initio and Informatica environments that converts their data flows into ones that can be leveraged by expressor. The company also has a variety of OEM relationships specialising in various vertical solutions such as those offered by Baecore Group for municipal governments and MetaAnalytix for the healthcare industry.

In terms of technical partnerships the company partners with Teradata, Microsoft, QlikTech, Progress Software, Attunity, BIReady, and Melissa Data. A number of these partnerships, for example those with Teradata, QlikTech, and Melissa Data, go beyond mere marketing and extend to technical integration. In the case of Teradata, the two companies have a long history of working together to make sure the expressor product scales and performs to the levels expected by Teradata customers. The companies also engage in joint marketing and selling.

A notable additional support feature is the expressor Community web site, which provides a forum, knowledge base, documentation and tutorials. It is planned that the Community will include an area where users can share domain-specific semantic types and extensibility options.

**expressor web address**: http://www.expressor-software.com

**expressor free product**: http://www.expressorStudio.com

**Community site**: http://www.expressorCommunity.com

**expressor Blog**: http://www.expressor-software.com/blogs

## Summary

The Community Edition has been very successful in raising expressor's profile since it was released in 2010. However, the company is now more interested in attracting the attention of companies that want a serious data integration product rather than hobbyists and others who just want to play with a free tool. Thus the company has introduced the Desktop Edition of its product as a "try before you buy" offering. This lacks the version management and multi-developer capabilities of the Standard and Parallel Editions and means that the Desktop Edition effectively provides an easy proof-of-concept at a low cost entry point for a single developer.

From a technical standpoint it is its semantics that makes expressor stand out. From a user perspective semantics is all about ease of use and expressor has extended its focus on ease of use both through its rules editor and its Extension Framework and packaged integrations. If ease of use is a major consideration for you (and it should be) then expressor is worth serious consideration. The other things you are likely to want or need are a low price and high performance. The former is built-in and with the Parallel Edition we expect expressor to be one of the best performing products in the market.

### Further Information

Further information about this subject is available from
http://www.BloorResearch.com/update/2117

## Bloor Research overview

Bloor Research is one of Europe's leading IT research, analysis and consultancy organisations. We explain how to bring greater Agility to corporate IT systems through the effective governance, management and leverage of Information. We have built a reputation for 'telling the right story' with independent, intelligent, well-articulated communications content and publications on all aspects of the ICT industry. We believe the objective of telling the right story is to:

- Describe the technology in context to its business value and the other systems and processes it interacts with.
- Understand how new and innovative technologies fit in with existing ICT investments.
- Look at the whole market and explain all the solutions available and how they can be more effectively evaluated.
- Filter "noise" and make it easier to find the additional information or news that supports both investment and implementation.
- Ensure all our content is available through the most appropriate channel.

Founded in 1989, we have spent over two decades distributing research and analysis to IT user and vendor organisations throughout the world via online subscriptions, tailored research services, events and consultancy projects. We are committed to turning our knowledge into business value for you.

## About the author

Philip Howard
Research Director - Data

Philip started in the computer industry way back in 1973 and has variously worked as a systems analyst, programmer and salesperson, as well as in marketing and product management, for a variety of companies including GEC Marconi, GPT, Philips Data Systems, Raytheon and NCR.

After a quarter of a century of not being his own boss Philip set up what is now P3ST (Wordsmiths) Ltd in 1992 and his first client was Bloor Research (then ButlerBloor), with Philip working for the company as an associate analyst. His relationship with Bloor Research has continued since that time and he is now Research Director. His practice area encompasses anything to do with data and content and he has five further analysts working with him in this area. While maintaining an overview of the whole space Philip himself specialises in databases, data management, data integration, data quality, data federation, master data management, data governance and data warehousing. He also has an interest in event stream/complex event processing.

In addition to the numerous reports Philip has written on behalf of Bloor Research, Philip also contributes regularly to www.IT-Director.com and www.IT-Analysis.com and was previously the editor of both "Application Development News" and "Operating System News" on behalf of Cambridge Market Intelligence (CMI). He has also contributed to various magazines and published a number of reports published by companies such as CMI and The Financial Times.

Away from work, Philip's primary leisure activities are canal boats, skiing, playing Bridge (at which he is a Life Master) and walking the dog.

## Copyright & disclaimer