Domain Specific Visual Query Language for HEP analysis
or
How far can we go with user friendliness?

Vasco Amaral*, Sven Helmer, Guido Moerkotte
Universität Mannheim, Germany

Abstract

There is a permanent quest for user friendliness in HEP Analysis. This growing need is directly proportional to the growing complexity of the Analysis frameworks' interfaces.

In fact, the user is provided with an Analysis framework that makes use of a General Purpose Language to program the query algorithms. Usually the user finds this overwhelming, since he is presented with the complexity of the intricacies of the systems. This way the final user of HEP experiments becomes a forced programmer or an application developer.

In our opinion this inflicts directly or indirectly in the query system performances. For this reason we have decided to invest in a new line of research. We aim to find a solution that balances the complexity and variability of the Analysis queries with the need for simpler query systems interfaces. The ultimate goal is to save time on query algorithms production.

We present how we explored the hypothesis of generating a visual query language specific for the HEP high level Analysis domain. The prototyped framework developed so far, PHEASANT, is giving us arguments in the feasibility of this approach. Therefore, like in any young Human Centric development project, this raises the need of a broad discussion in order to validate it. We believe to be opening an new fruitful research topic among the community and we expect to motivate both computer science and physicist experts into the same discussion.

MOTIVATION

Present high level Analysis frameworks provide the user with two communication channels. In this paper the result visualization facilities are called the query **output channel** (please do not confuse with the data input/output channel). To flexible querying the information systems, the frameworks provide General Purpose Languages (GPL), like C++, C or others (see Fig.1). We call it query **input channel**. In our opinion, while in one side the **output channel** has improved pretty much, on the other side much is still to be done in the **input channel**.

In fact, these Analysis frameworks, specifically designed for this domain, do not detach the libraries of functions from a GPL. With time, the libraries become larger and more generic. The consequence is that the usability decreased because of the multiplicity of entry points, parameters and options offered to the end user that is now requested to increase his programming skills.

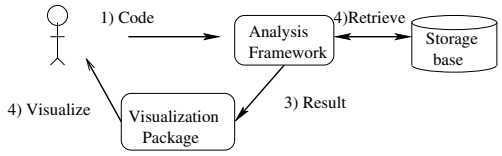

Figure 1: Nowadays approach in HEP - textual Input channel and visual output

While in one hand the flexibility of GPL brings great expressive power to the Physicist experienced in programming, on the other end, it fails to present a simple language for the non-experienced user that has to code to work on his Analysis. This situation reflects immediately in the time spent to code, compile and debug the query application developed, plus the possible inefficiency of running the non-optimized application in the Analysis frameworks shared resources. As in any software application the way end users interact with databases to access data can have demolishing performance effects. The motivation of our work is to develop and find better and new ways to optimize the process of HEP Analysis. The way we propose to do that, is by introducing a domain specific visual query language in order to improve the **input channel** by improving the learning curve without loosing flexibility and expressivity.

The following description will specify further this topic and report our research.

LINE OF RESEARCH

Based on our experience [8, 9, 1] with a real running experiment, Hera-B, with real Analysis data and users, we started our research by gathering the data model and query patterns.

As we have observed, Analysis tends to have unpredictable queries, meaning that instead of a limited set of queries we have gathered a wide variety of them. However being different, we have noticed that it was possible to infer certain standard commonalities (meaning common structure) and predicatable variabilities (meaning details that make each query different from each other[7]).

* researcher at LIP

Since the user is frequently presented with different kinds of Analysis frameworks, each one with different types of coding paradigms and data representation, we have tackled the problem by introducing a new layer of abstraction that hides all those coding details. By different coding paradigms we mean: iterative, functional or object oriented. By different data representations we mean that the user deals with objects or with table like structures named N-Tuples.

This new layer provides an unifying vision of the different Analysis frameworks. It presents the user with a flexible query language that, in opposition to a general purpose language, deals with concepts (object concepts and operations) of this specific domain (Domain Specific query Language). As we have already mentioned, the traditional procedure for querying the Analysis framework uses the **output channel** as Visual (under the form of histograms), and the **input channel** as textual. From the Computer Science, we know that Visual Languages have significant advantages over the textual approaches, specially for unexperienced users. Visual systems are a way to enlarge the user-machine channel. Therefore, our proposal was to build a Domain Specific Visual Query Language (DSVQL)[3], see Fig.2.

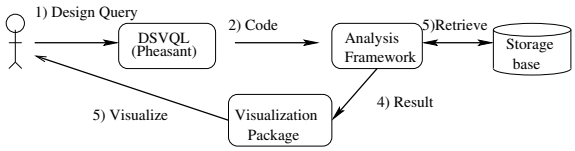

Figure 2: Our proposed approach - Visual input and output channels

The requirements for this DSVQL are to make use of HEP Analysis concepts and give the user the control on designing/modeling his query in a visual way. In consequence, the visual system is then able to map the query into the target Analysis framework code, which on its turn is very particular for each experiments.

A typical DSVQL is reported to be a R&D project on itself, with significant implementation effort. Nevertheless, benefits are huge since by raising the modularity and abstraction layers the optimization potential is increased. The burden of efficiency is no longer in the user's side but on the experts' one. There is a better control over the query patterns, since the DSL is much more restricted then GPLs, which means that developers can track more easily the bottlenecks.

In order to validate this proposal for approaching the query **input channel** and test its feasibility we have designed a language, called PheasantQL[4, 10], and we have implemented a small testing framework (named Pheasant prototype[10]) to learn about the unclear implementations details. We believe that this elegant approach pushes for the user friendliness side of the Analysis frameworks and increases the abstraction from the code world.

We will next describe our procedure to design the language, and following that how to implement a prototype to deal with it.

Language definition

Any query language should be specified by means of a formal syntax and semantics. This approach is beneficial since by doing so we are forced to develop both major concepts of the language and the details, leading to a truthful implementation. Additionally, the user has a unique and clearly determined semantics for any sentence in the language.

The syntax of a language is a set of rules that define the ways symbols may be combined to create well-formed sentences in that language. The semantics, on the other end deals with the meaning of programs, which means how they behave when executed on computers.

In [4, 10] we specify the Pheasant language. We introduce the syntax with the notation and alphabet of our proposed language. Following that, we specify the semantics of the language making use of translational semantics. In other words we define the semantics of our language by mapping it into the very well-known (for Computer Scientists working with core database technology) intermediate Object Algebra whose description can be found for instance in [12].

Prototyping

Due to the space limitations we can not dive into the details, so we just give a simplified overview. For further details we recommend [6, 11, 10].

The system was designed to cope with the several query transformation phases required to deliver a target query source code that should be compiled and run against a specific physics storage base.

We have devised three main modules, as seen in Fig.3: user interface, plan generator and code generator.

The User Interface is responsible to deal with the user's query edition, interactively notifying the user of incorrect syntax, see Fig.4. Internally a Concrete Components Graph is maintained and simultaneously mapped, using the observers pattern, to a corresponding Abstract Syntax Graph.

The Plan Generator starts by interpreting the Abstract Syntax Graph and transform it into an Abstract Syntax Tree, (which is easier to deal with by having simpler walk-down algorithms). It continues then by running an algorithm that walks down the AST and generates the corresponding algebraic plan like in Fig.5(for complete details please consult [4, 10]).

Finally, the query Code Generator looks at the algebraic query plan, optimizes it at the algebraic logical level and generates the physical operators. In the sequence of that, a new algorithm generates the required source code to be compiled and run against the storage base. This module is strictly bound to the specific target framework. The query code generator is implemented as a plug-in to our Pheasant framework specific to the Hera-B Analysis framework (in

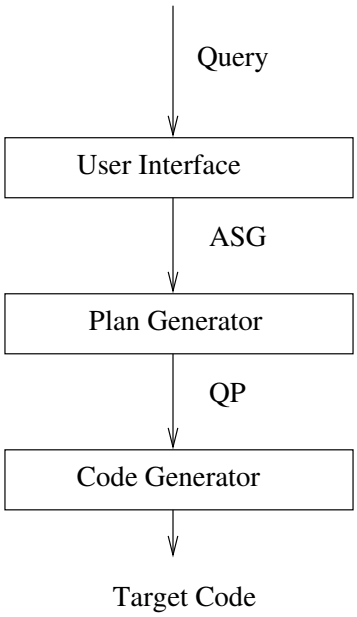

Figure 3: Different architectural layers

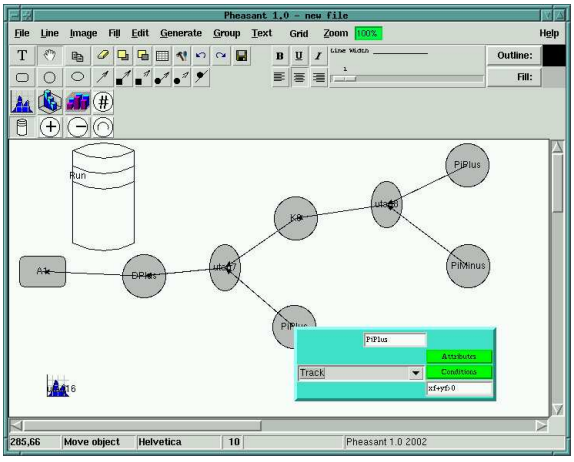

Figure 4: Pheasant prototype layout

this case BEE[13]). Other plug-ins can be added to deal with different target physics frameworks without necessarily impose changes to the rest of the query generation modules.

Usability evaluation

To support our claims that through our methods we manage to improve the efficiency, reduce the error rate and have a fast learning curve, we have to exercise a complete and non-biased evaluation of our language comparing it to a real life programming Analysis framework. In [11, 10] we explain how we have evaluated the usability in its three concepts of the implemented prototype, and that should be seen as guidelines for future framework evaluations.

We measured **effectiveness** to determine the accuracy

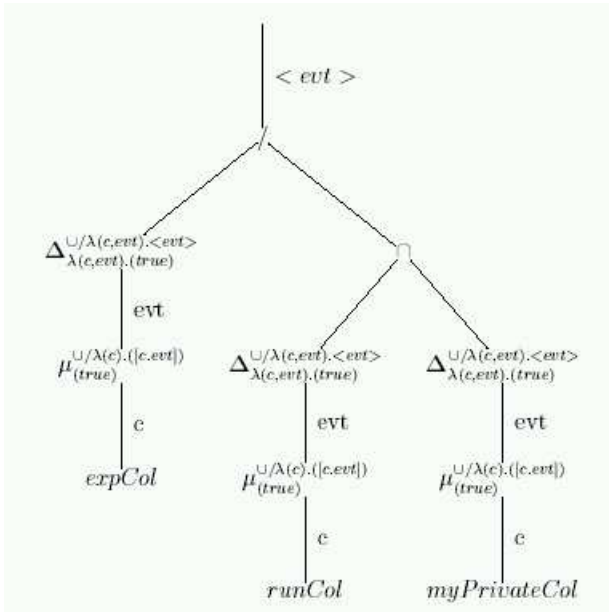

Figure 5: Example of a query plan from a simple query. For explanation and details consult [4, 10]

and completion when performing queries. We also observed **efficiency**, as measurement related to the level of effectiveness in time spent to complete a query. (Efficiency can also be measured in terms of the expense of various resources, such as mental and physical effort, time, financial cost, etc.)

We measured **satisfaction in use** to evaluate if the tool was free from inconveniences and generated positive attitude towards the use of it. In other words we measured how comfortable the user felt while using the system.

Although just dealing with a prototype that was not a full fledged implementation, since it was meant to test some implementation details, the results were very promising and gave us the confidence of being in the right track [11, 6, 10].

SUMMARY AND FUTURE WORK

We have proved the feasibility of the idea of a Domain Specific Query Language as a way to boost the user friendliness in HEP Analysis. As in any Human-Computer interaction system, the next steps will naturally involve a broader discussion in the research community in order to achieve the necessary improvements and extensions during the next product's life-cycles.

The query language itself should be subject to a broader discussion in order to evolve to the necessary expressive power.

Developers (Computer scientists, Software engineers, others) can now concentrate on the several architectural levels abstracting the physics, while the end users have a now a tool that prevents them from having to code for the most patternized queries as possible.

Some extra directions could be taken, for instance the in-

tegration in projects linked to the applicational level of the GRID should be studied and its impact in the productivity should be evaluated. A further point we want to work on (and which involves all abstraction levels) is to deal with "environmental" data, (meta-data), such like the introduction of query versioning and storing. This means, that users can save, load, and modify different versions of the same query. In this context we also think of storing answer sets temporarily for further querying.

We believe to be touching a very exciting topic of research and we challenge others to follow us and collaborate with us in this effort to increase the final users' productivity.

ACKNOWLEDGMENTS

This work was partly funded by the Portuguese Governmental Foundation of Science and Technology FCT in form of a Phd. scholarship ref. SFRH / BD / 8918 / 2002 to Vasco Amaral. We acknowledge also LIP (Laboratório de Instrumentação e Física Experimental de Partículas) for its support.

REFERENCES

[1] V. Amaral, A. Amorim, and et.al. Operational experience running the Hera-b database system. In H. Chen, editor, *Proceedings of CHEP 2001, International Conference on Computing in High Energy and Nuclear Physics, Beijing, P. R. China*, pages 396–397. Science Press, September 2001.

[2] V. Amaral, S. Helmer, and G. Moerkotte. Designing and implementing a new abstraction layer to optimize the HEP analysis process. *IEEE Conf. Record of Nuclear Science Symposium NSS, Portland, OR, USA*, pages N26–104, October 2003.

[3] V. Amaral, S. Helmer, and G. Moerkotte. A Domain Specific Visual Query Language for the High Energy Physics environment. In J.-P. Tolvanen, J. Gray, and M. Rossi, editors, *3rd Workshop on DomainSpecific Modeling, An OOPSLA 2003 Workshop, Anaheim, CA, USA*, pages 9–16. Jyväskylä University Printing House, Finland, October 2003.

[4] V. Amaral, S. Helmer, and G. Moerkotte. PHEASANT: A PHysicist's EASy ANalysis Tool. *Technical Report of the University of Mannheim: 8/03*, 2003.

[5] V. Amaral, S. Helmer, and G. Moerkotte. A visual query language for HEP analysis. *IEEE Conf. Record of Nuclear Science Symposium NSS, Portland, OR, USA*, pages N26–105, October 2003.

[6] V. Amaral, S. Helmer, and G. Moerkotte. PHEASANT: A PHysicist's EASy ANalysis Tool. In J. Carbonell and J. Siekmann, editors, *LNAI Lecture Notes in Artificial Inteligence*, pages 3055:229–242. Springer Verlag, June 2004.

[7] V. Amaral, G. Moerkotte, A. Amorim, and S. Helmer. Studies for optimization of data analysis queries for hep using Hera-b commissioning data. In H. Chen, editor, *Proceedings of CHEP 2001, International Conference on Computing in High Energy and Nuclear Physics, Beijing, P. R. China*, pages 154–155. Science Press, September 2001.

[8] A. Amorim, V. Amaral, and et. al. The Hera-b database management for detector configuration, calibration, alignment, slow control and data classification. In I. P. Mirco Mazzucato, editor, *Proceeding of CHEP 2000, international conference on Computing in High Energy and Nuclear Physics, 7-11 February, Padova-Italy*, pages 469–472. Imprimenda, Padova, Italy, February 2000.

[9] A. Amorim, V. Amaral, and et. al. The Hera-b database services for detector configuration, calibration, alignment, slow control and data classification. *Elsevier Science, Computer Physics Communications*, 140(15):172–178, October 2001.

[10] V. Amaral Optimization of data analysis queries for massive data-volume HEP Information Systems. *Phd. thesis, University of Mannheim, to be published by end of 2004*.

[11] V. Amaral, S. Helmer, and G. Moerkotte. Engineering a New Abstraction Layer to Optimize the HEP Analysis Process. *IEEE Transaction in Nuclear Sciences Journal (TNS), ISSN: 0018-9499, CODEN:IETNAE, volume 51, number 4, August 2004, pp.1441-1448*

[12] L. Fegaras and D. Maier. Optimizing object queries using an effective calculus. *ACM Transactions on Database systems*, 25(4):457–516, December 2000.

[13] T. Glebe, Clue - The BEE event model library. *HERA-B Note 01-138, Software 01-019, DESY, 2001*

[14] V. Amaral Project Pheasant website: *http://pi3.informatik.uni-mannheim.de/\~amaral/pheasant*. *http://cern.ch/vasco.amaral/pheasant*.