Use of CASE tools in the Design of a Data Flow Protocol for the RD13 Data Acquisition System.

G. Ambrosini¹, C.P. Bee², S. Buono², R. Ferrari¹, Fumagalli¹, P. Ganev², S. Hellman², J. Harvey², R. Jones², A. Khodabandeh², L. Mapelli², G. Mornacchi², P. Palazzi², G. Polesello¹, D. Prigent², C. Rondot², F. Tamburelli²

¹: Dipartimento di Fisica dell’Università e Sezione INFN di Pavia, Italy
²: CERN, Geneva, Switzerland
3: Centre de Physique des Particules de Marseille, IN2P3, France

The Computer Aided Software Engineering (CASE) technology is very powerful for complex software developments like the one we will have in the future LHC. We can not continue to produce and maintain "by hand" software that takes years to write and should last for decades. One of the RD13 projects goals [1] is to study the CASE technology adequacy to the LHC Data Acquisition (DAQ) software production. In order to do this we have chosen to apply and evaluate different methodologies - supported by tools - to the same problem: the data flow protocol of the RD13 DAQ system. The objective is to perform all the phases of the development following a methodology. This means that the analysis, design, prototyping, implementation and maintenance should be done with a CASE tool.

1 Introduction

In order to perform a full scale evaluation we have chosen to base our study on a real project, with all the constrains that this means. In our case the project is the data flow protocol for the RD13 DAQ system. A detailed specification of the project can be found in [2] and [3]. To make our evaluation more accurate we decided to compare different methodologies that are supported by commercial CASE tools - StP and Artifex. But each time the test project is the same. The goal is to see how much a CASE tool can bring to software production in data acquisition system? To what extend a CASE tool can improve the analysis, testing, design, implementation, maintenance and documentation? What are the limitations and the dangers?

2 StP

The first CASE tool we have chosen is called StP [4] (Software through Pictures) by IDE (Interactive Development Environments). Stp is a set of CASE tools supporting analysis and design methods for software production. It offers diagram editor tools for data flows, data structure, structure charts, entity relationships, control specifications, control flows and state transitions. There is also a picture editor and object notation editor which allow the user to associate a set of properties and values with any objects in any editor diagram. A document preparation system allows designs to be printed in many formats.

To model the protocol we have used a real time structured analysis method, as described in Ward/Mellor [5] and Hatley/Pirbhai [6]. Both variations of the method are supported by StP. The model is defined, specifying what the system must do in terms of hierarchy of control and data flow diagrams going from the context diagram down to the specifications. The application data base and all the other data structures used to manage the protocol are defined according to the entity relationship data model, or the equivalent data structure. The protocol itself is specified in terms of state transition diagrams. These different modeling tasks are performed by
means of the corresponding StP editor, the results are then stored in the StP data dictionary and are available both to the StP editors and to user defined modules - e.g. code generator.

We had to overcome [7] some of the StP deficiencies:

- Since a protocol is basically a state machine, the decomposition in state transition diagrams would make the definition clearer. Unfortunately StP does not support this feature. We made a special coupling of data flow and state transition diagrams that allowed us to use the DF nesting to define the missing ST hierarchy.
- More over, a code generator not supported by StP would be a great help to make the Analysis Design Implementation cycle seamless. We used the above method and the data dictionary openness of StP to specify a code generator.

At this stage of the project we decided that it would be interesting to redo the exercise with another tool which supports another method but has the features that we found missing in StP.

3 Artifex

The second methodology/tool that we are currently testing is called PROTOB, an object-oriented methodology based on an extended data flow model defined using high level Petri Nets, called PROT nets, and its associated CASE toolset called Artifex [8], by Artis S.r.l.

Artifex consist of several tools supporting specification, modeling, prototyping and code generation activities within the framework of the operational software life cycle paradigm. Artifex fully supports system analysis, design, simulation and implementation through the same GUI.

- The analysis and design phase is done using the graphical formal language for the high level concurrent Petri Nets, that integrates sections of C and Ada code.
- The simulation generation and execution is done with two buttons only. During this phase you can simulate, set break points and step through your concurrent task model.
- The emulation supports distributed code generation from the same model that was used for analysis and simulation. During emulation visual monitoring of the Petri Net is possible using the same GUI used for the two previous phases.

The protocol is specified in terms of High Level Concurrent Colored Petri Nets [9]. This model can be seen as a combination of the data flow and state transition of the SASD methodology. The object model of Artifex allows independent module definition, testing and reuse.

A major extension to the Artifex environment is Quid, a database definition and query system. We intend to use Quid for the database aspect of the DAQ protocol we designed. But Artifex can be combined with any other commercial (Oracle) or home made (ADAMO [10]) data definition system.

We have finished implementing and porting the protocol to the target with Artifex, and we are starting performance measurement.

4 Comparison

The most important outcome of the exercises is our knowledge on CASE technology in general and in regard to DAQ protocol design in particular. Here are some conclusions out of the comparison of the two methodologies/tools.

Methodology: This could be a trivial remark but we think it is basic that a CASE tool supports a methodology. Otherwise it turns itself into a very expensive drawing tool.
In our case StP supports SASD with Real Time extension [5][6]. But it also allows you to customize the methodology to your needs.

Artifex supports only the High Level Concurrent Colored Petri Net [9] which is not at all a limitation.

**Support of the methodology:** It is fundamental that the tool provides a good interface - if possible graphical - to the methodology. This is done by both tools, but in Artifex the viewing feature that does not exist in StP, is a great help in cutting down the diagram complexity.

Another feature of the diagram editors of the two tools that we found fundamental is the decomposition facility that helps to do a step by step refinement of the model from the top level down to the specifications.

- StP supports this in its data flow editor according to the RTSASD methodology but does not provide anything equivalent for the state transition diagrams.
- Artifex implements transition decomposition through the objects and the viewing feature.

The tool must also check the consistency of the user design against the methodology. We distinguish two types of checking: on-line, done directly by the GUI, and off-line, started by the user.

- Both StP and Artifex have the two types of checking. But StP has a stronger off-line checker, and Artifex a stronger on-line one.
- It must be also stressed that the Artifex design is executable. It is meant to be simulated and used in code generation. This adds to the constrains by being an implicit checking.

**Life cycle management:** Another matter of concern in CASE technology from the point of view of the user is how much of the life cycle does the tool cover? In other words, what does the tool do for me and what should I do by hand? Given the following chronological steps: Analysis, Test&Simulation, Design, Code Generation and Maintenance we expect the tool to support them in a seamless manner from the GUI down to the code.

- StP covers analysis and design phase the corresponding consistency checking. But the connection of the two is done by hand and no logical checking of it is performed by the tool. Moreover they provide neither a simulation facility to validate the protocol, nor generates code, but only SQL code to create database tables, and C header files.
- Artifex supports analysis and design in such a seamless manner that it is hard to distinguish them. We used the simulation tool to validate the dynamic of the protocol. We found the code generator very useful to help us to concentrate on the logic instead of the syntax. All those steps are well integrated in the life cycle using the same GUI, which makes the maintenance automatic and the overall process very pleasant.

**Learning/Using:** The tool must be easy to learn and to use. If the methodology is too rebarbative or the interface too complicated the tool will not last long. Both StP and Artifex follow the rule. Moreover they come with courses provided by company experts. A general principle that we noticed is that in order to get the best out of a course, one had to be already familiar with the tool and know about the methodology.

**Document Generation:** In the project development a critical point is the review meeting where physicists and computer scientists agree on what is needed and what should be provided. It is then important that the two groups talk the same language. A document prepa-
ration is then a very efficient way of producing paper documents that are understood, discussed, and signed by everybody. Stp and Artifex have both a very good customizable document preparation systems.

Openness: The openness and customizability could be interesting features depending on the type of application. But they should not be substitutes for other features.

- Stp uses a Data Dictionary, to store its diagrams, which is accessible from the user code using a query language (OML). We used this to specify a code generator.

- Artifex also is open but the other way around. The outside world is freely accessible from Artifex. This is how Artifex can be coupled with data base CASE tools.

Run Time Libraries: Some of the tools, mostly the ones that generate code - e.g. Artifex - rely on a run time library provided with the tool. During the past few years a tremendous effort was undertaken by the software community to reach a high level of platform independency. One should be then careful to not to become CASE dependent. We feel that the run time library dependency of the tool is an important point to take into account before investing in a CASE tool.

5 Conclusion

After the two experiences we believe that there is a great future in CASE technology in general and in HEP large software production in particular. The benefits in project management, life cycle, code reliability and maintainability are tremendous. Improvisation is no longer possible for ten years development and ten years life time software. Now that the community is moving toward structured and methodological software production we should not move the anarchy to the CASE level. That is why we think that one should be careful in the choice and use of a tool by considering parameters such as the ones mentioned above. The number of unexpected but still important problems that we met during our work with CASE tools, taught us that R&D in this field is an investment.

6 References

1. L. Mapelli et al., A Scalable Data Taking System at a Test Beam for LHC, CERN/DRDC/90-64/P16, CERN/DRDC/90-64 Add.1, CERN/DRDC/90-64/P16 Add.2, 1990.
3. G. Ambrosini et al., The RD13 Scalable Data Acquisition System, These proceedings.
10. M. G. Green, The ADAMO Data System, An Introduction for Particle Physicists, Royal Holloway and Bedford New College, RHBNS 89-01.