TS2P an open source testbed & simulation software platform

E. Noulard, Y Dufrenne

TS2P an open source testbed & simulation software platform

E. Noulard[1], Y. Dufrenne[2]

1: BT Consulting & Systems Integration, eric.noulard@bt.com, http://www.btconsulting.com
2: EADS Astrium, yves.dufrenne@astrium.eads.net, http://www.astrium.eads.net

Abstract: Integration of open source software in industrial environment and process may be something critical. We present hereafter the practical experience of an industrial open source project in the domain of real-time testbed and simulation. A proposition is made to build such a particular project experience and to develop an open source software platform for testbed and simulation.

Keywords: open source, real-time, simulation, sampling protocol, TSP.

1. Introduction

Ten years ago we were still informed of the arrival of the envisioned open source software wave, with software such as the web centric software like Apache [2], general purpose operating systems like Linux, and new development model [1]. At that point industrials were considering open source software like toys or at most like prototyping tools but it was clear for them that they could not be used within industrial application.

Nowadays the worldwide internet connectivity and the dramatic broadband accessibility growth [3] made it possible to change the open source software (OSS) status: the OSS are now present in industrial projects, and it seems they are here to stay. Nevertheless, there are many ways for industrials to integrate the OSS in their daily business; ranging from simple use as a particular case of OTS (Off The Shelves) Software to full Open Source supporter such as OSS project founder or driver. IBM with the Eclipse Framework [4], Open Cascade with the Open Cascade Software [5], and the LEON Open Spatial processor from ESA [7], to name just a few of them.

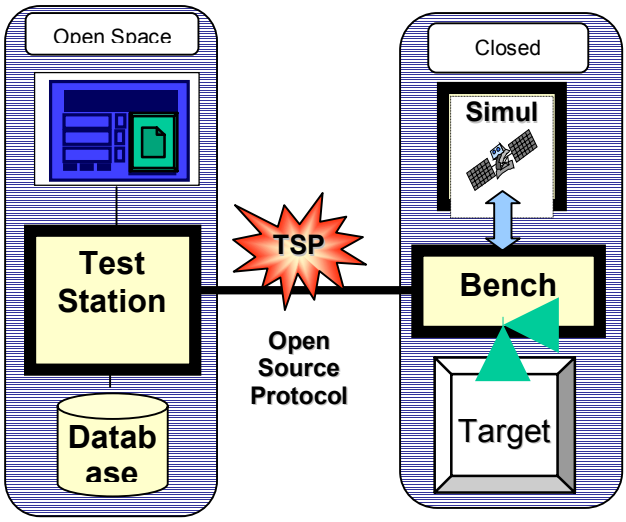
Exhaustive presentation of Open Source software, hardware [7], philosophy or even business model is out of the scope of this paper. We focus on the practical example of an industrial open source software and project with its business model companion. The interested reader may find broader information starting point on OSS in [3] or [6].
We present hereafter the industrial experience of a niche market OSS project and software [6] that lead us to a proposition for a wider industrial open source platform for testbed & simulation associated with a new industrial service-oriented open source business model.

2. An industrial open source project

2.1 Genesis

BT C&SI and EADS Astrium have been working together for several years in the domain of the validation testbed for satellite assembly, integration and test.

Both companies do have firm technical experience in the domain of real-time test beds. One of the recurrent features of those testbeds was the need to observe or "sample" evolving data (simulated physical element or spied hardware). Some solutions already exist (internals, COTS, …), but were too specific, heavy, or complex.



And so appeared the idea of a versatile real-time sampling protocol. The protocol and its implementation should be reusable between different projects in order to lower the cost. This protocol could become a mighty silver bullet only if it makes easier the separation of concern (e.g. monitoring GUI versus real-time simulation, whoever is in charge of it).

This implies that the usage of the protocol must be free. Then every company developing a solution which needs such protocol may simply use it.

2.2 TSP: first step in the Open Source World

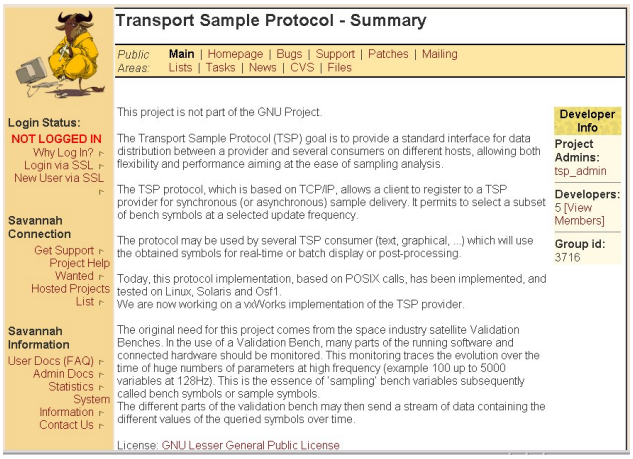
In august 2002, BT C&SI and EADS-Astrium decide co-specify the future Transport Sampling Protocol (TSP) and to implement the C library version. At this point the project is a classical customer (EADS Astrium) – software provider (BT C&SI) contract. The two parties then discover the benefits to open source the specifications and the code, in order to try external re-use and improvement through an open source project. The main goal is to provide better re-use and sub-contracting through an open "standard" way to observe evolving data in real-time testbeds.

After managing several problems and human factors, the two companies set the seal on an open-source project specifying how they will proceed.

The open source project is submitted on Savannah on December 2002 and accepted some weeks later:

http://savannah.nongnu.org/projects/tsp

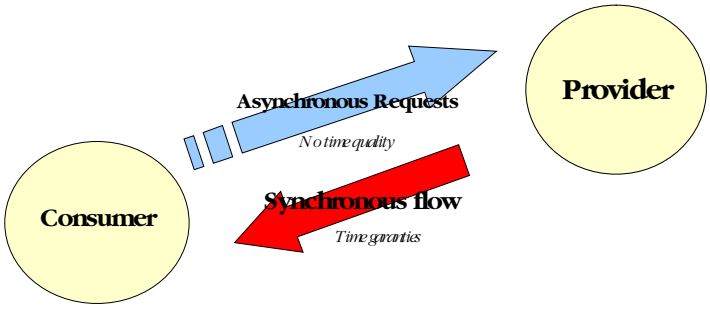
is born.



2.3 TSP at a glance

The TSP may be used to efficiently sample and observe data in either real-time or not simulators and testbeds. The TSP architecture follows the publish/subscribe paradigm [9] in which a data producer is called a TSP provider and a data consumer is called a TSP consumer. Even if the TSP shares the publish subscribe concept of DDS [10], the TSP is much simpler than DDS. Moreover the TSP focus on the observability and the on-the-wire portability of the data link, which is not the case for DDS. DDS is a data-oriented middleware service, including high level constraints (real-time, QoS, Ownership).

The TSP protocol is networked and multi-consumer/multi-provider offering a wealth of data collection possibility. The TSP consists in an open source development toolkit available in portable C and Java which make TSP readily available on several platforms.



2.4 TSP: industrial experiences

The Open Source TSP has since been used on three real-life projects for three different industrial partners for either EADS-Astrium or BT C&SI.

Each time a project using TSP needs a new feature, it has to fund the improvement, to design it and to implement it. Afterwards, the useful generic part is contributed back into the open source project. Other projects using TSP may get those evolutions smoothly when needed.

TSP is now a small but useful open industrial project used directly or indirectly (through subcontracting) by at least four big companies in space or telecom industry in five different projects.

3. Industrial Open Source

BT C&SI would like to generalize the TSP process thanks to the technical and business success of the TSP project. We propose a new open source business model with an open source platform for testbeds and simulation.

3.1 Industrial needs

Industrial companies (telecom, satellite, transport, military) developing products want to reduce the cost of software development and manage to deliver the right working solution on time.

Those industrial products usually follow the life cycle steps:

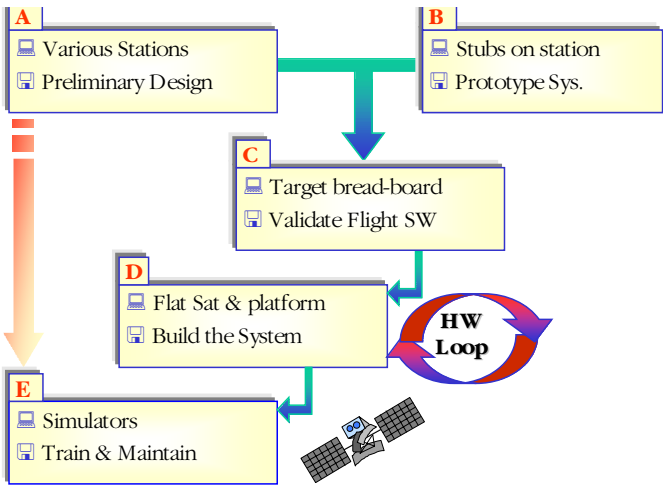
A) Preliminary Design

B) Design/Study/Prototype

C) Validate by bricks

D) Integrate and Qualify

E) Maintain



This is true for satellite, telecom systems, aircraft and probably many other industrial technical products. During steps B, C and D one needs to build some kind of software and/or hardware simulation testbeds specialized for the concerned business domain.

3.2 A sample industrial process

Let's focus on a satellite production example.

Step B consists in the satellite system design which implies the use of a functional simulation testbed. This kind of testbed is used by AOCS experts to validate and tune the on-board and ground system. The testbed may be software only or may include some characterization features used to evaluate particular satellite hardware equipment (GPS receiver, star tracker, gyroscope...). It usually includes simulated hardware processor components used to co-design the system, such as a FPGA simulated Leon processor [7] or a software processor emulator. This kind of simulation testbed are generally not real-time but as-fast-as possible in order to efficiently run many test cases.

Step C consists in the validation of the designed system. This step usually involves more hardware and much more software whose purpose is to achieve full closed-loop simulation. The main objective is to validate on-board software/hardware and their expected performances. The hardware missing in the (closed-)loop should be simulated sometimes at bus of electrical level. The included hardware should be stimulated in order to fake the missing part of the *real world*. Those real parts must be stress-tested in order to verify their functional range is the one assumed during design. Those simulation testbed usually are hard real-time ones. A satellite validation testbed may be very complex since it may be used for global systems simulation as rehearsal means for the to be launched system which includes operational ground segment.

Step D consists in the satellite system assembly, integration and test (AIT). Ideally the previous step C validation testbed is reused, sometimes in open-loop mode, in order to incorporate and test final hardware (and embedded software) in the system currently being assembled.

Step E is when satellite systems has or will entered soon operational state. Satellite is now in space, either fully operational or in orbital transfer phase. The previously built testbed are used to:

- Train operator
  Operator cannot exercise on a real satellite system. Classically a real ground-segment is connected to a simulated space segment (a simulation testbed).
- Solve problem occurring on the real system
  When problem occurs in space, the validation (and/or integration) testbed is used to reproduce the problem (if it's possible) on the ground. When reproduced, a new on-board software is created and validated using the simulation testbed before uploading the new software in space. In this situation the simulation testbed may need further development if a non anticipated test case shows up.

This satellite system example should share common properties with other industrial systems (telecom systems, aircraft, car...), so the previously described steps, including the issues should be similar.

3.3 Some common industrial issues

If you consider the software development part of such technical projects, some technical choices should be made in order to cope with following recurrent issues:

1. Costly start-up time on specific proprietary environment

   When a new person comes in a specific simulation testbed development team, start-up time is on average 1,5 man/month. This start-up delay is mostly due to the complexity of simulation testbed and their proprietary nature.

2. Software knowledge management problem

   Even a software standard based simulation testbed contains a lots of in-house software mostly developed by subcontractors. When the product is out and/or going into maintenance phase the industrial may come up against difficulty to maintain the sufficient knowledge. Since the simulation testbed may/will be used in maintenance over a long period (20+ years maintenance for telecom satellite system), this may become a real industrial issue.

3. Software reuse problem

   Large industrial companies usually built several simulation testbeds a year in different (unrelated) projects or department. Internal cross-project reuse is not as easy as it seems and revalidation cost may be paid by several projects, even if they potentially could have shared those costs.

4. Control software vendor license policy

   When using technical niche market COTS, industrial is sometimes bound to a couple of software providers which may be bought by competitors. This situation sometimes induce unpredictable software licenses cost raise during maintenance phase.

Note that those issues are *not the same* as general purpose software product (widespread langage compiler, word processor, web servers) where normalized standard and multiple vendors solutions may tackle those problem right away.

Inside the specific testbed & simulation market there is no such ready-to-use solution and it won't probably be one any soon, because of the specific nature of each project.

There exists tools, but they usually are not the big part of the testbed. As an example, today the simulation testbeds are getting more and more distributed. In this area you have tools (CORBA, HLA, DDS,...) but even if they provide necessary building blocks they are not “solution”. Moreover, if you did bought a commercial DDS product for a large project it may be out of budget for a smallest one, due to the licence cost.

3.4 Open source software answers

Open source software (OSS) may tackle some of those issues, more easily than proprietary do.

We should clearly and early point out that OSS does not mean free of charge and that OSS may come with full commercial support solutions which are necessary in industrial environment. A simple web search with “*professional open source support*” should convince the reader. You may look at open source initiative [6] in the section “*O.S. for Customers, Business, and Programmers*” in order to find more information on the subject. Nevertheless, we will come back to those “professional” aspects when introducing the TS^{2}P business model.

The OSS industrial use may tackle some of the previous industrial issues and derive the following benefit from OSS:

A) Accelerate software development

   1. Internal and external software reuse: anyone is free to use it and to give contribution back
   2. More efficient training since sub-contractor of different industrial projects may train their team efficiently at lower cost since software may be used for free training.

B) Reduce cost

   1. Concentrate on core business and open source non business critical software
   2. Stimulate competitiveness because anyone wanting to acquire the knowledge may do it (at its own cost).

C) Ensure long term stability

   1. Interoperable [de-facto] standard
   2. Open industrial work group

Those benefits may be shared by almost every open source software, including general purpose ones. Though, the efficient industrial use of OSS in the domain of simulation testbed may need little more formalized recommendation in order to guarantee success.

4. TS^{2}P: Open Source platform project

In March 2005, BT C&SI proposed to its main customers to get involved in the TS^{2}P project: the **T**estbed & **S**imulation **S**oftware **P**latform.

4.1 TS^{2}P: the content

The platform will contain independent software modules with TSP as the first one. The stakeholders will put in, software modules which are generic enough to be useful and do not entail their competitiveness. Everybody can reuse the software

building blocks of the platform, but development of one module is leaded autonomously by the people in charge of it (preferably most contributing people).

Software modules may be development toolkit, standalone applications, GUI, etc... Modules may be linked with each other or not.

Every module should adopt an open source licence such as GNU GPL, GNU LGPL, Apache or BSD. The license must, at least, enable the use of the module in industrial and commercial context. Licences for documentation could be open too just like GNU FDL or Creative Commons Sharealike.

The TS^{2}P steering committee will vote the new module introduction. A software module may not endanger the specific business of a stakeholder, so only general test and simulation modules should enter the platform. A software module not used by any project (including maintenance) for more than 2 years shall not stay in the platform.

4.2 TS^{2}P: the stakeholders and their role

Each software module manager handles its module autonomously. Each module manager has a place in the steering committee of the platform. The steering committee only deals with “platform-wide” decision like module addition or deletion.

Every stakeholder should clearly promote the use of the platform in its own projects for which a simulation testbed is used or built. It does not mean that one is forced to use it but it *should appear as a valuable choice in the invite bids* when appropriate.

This promotion is not a marketing one but the only concrete means for the stakeholders to ensure the life of the platform.

4.3 TS^{2}P: support cost

The stakeholders *do not support the cost* of the platform but instead all cost should be taken from project using the platform. This does not preclude pure sponsoring, but this is not the preferred way to support the platform.

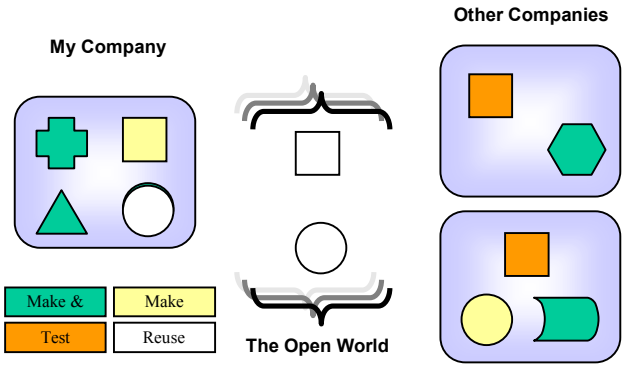
The TS^{2}P must be **project driven**.

4.4 TS^{2}P: an open source business model

The business model is totally service oriented:

- NO software product would be sold from the platform. The platform will be, as TSP, essentially project driven.
- Evolution of TS^{2}P is led by project. Instead of buying custom software the users will order service to include, enhance and maintain TS^{2}P modules.

The great difference with typical custom software service buying is that the open platform will leverage the cost between unconnected testbeds and/or simulation projects.



Note that the proposed TS^{2}P business model is different of other existing business models:

- TS^{2}P is not a product like JBoss [11,12] or MySQL [13] and their associated service around OSS product business model
- TS^{2}P stakeholder will not package existing OSS like linux distributors do (RedHat, Mandrake, SuSE/Novell)
- TS^{2}P will not support and/or develop OSS for marketing argument as some software companies do.

TS^{2}P stakeholder will use the platform for practical and industrial reasons.

4.5 TS^{2}P: recommended business loop

The TS^{2}P ideal business loop is:

- When inviting for tenders one indicates clearly that TS^{2}P should be used when appropriate
- When examining bidders’ proposal, one chooses the best one as usual, but TS^{2}P objectives are incorporated in the other classical criterion choice (price, technical validity etc...)
- If a project will use TS^{2}P, during project kick-off one identifies (re)usable TS^{2}P modules and potential new modules that could comes in during the project
- At project ending ensure that evolution of TS2P modules and/or new modules will be contributed back to the platform.

4.6 TS^{2}P: a concept to build together

Fifteen people of 6 companies attended the first TS^{2}P workshop and declared to be both surprised and interested in such a process. The TS^{2}P project is in its infancy but building on the TSP experience it should be successful.

5. Conclusion

We have drawn experience from a niche market open source project and propose collaborative open source software platform for testbed and simulation. We think both the TS^{2}P project and the associated business model will prove itself useful for industrial interested in the domain of simulation testbed.

6. Acknowledgement

The authors acknowledge people at BT C&SI and EADS Astrium who decided at the beginning that such an industrial open source project was possible and in particular José Dekneudt and Jean-Paul Chaumier. Special thanks go to the worldwide TSP development team for its valuable contribution for the past 4 years and the forthcoming ones.

7. References

[1] Eric S. Raymond: "*The Cathedral and the Bazaar*", http://www.catb.org/~esr, 1997.

[2] Apache Foundation: "*Apache HTTP Server*", http://httpd.apache.org/.

[3] Diane Revillard: "*Livre Blanc, Organisations et Logiciels Libres*", http://www.diemark.net/, 2005.

[4] Eclipse Foundation: http://www.eclipse.org/, 2005.

[5] Open Cascade S.A. http://www.opencascade.com, 2005.

[6] Open Source Initiative (OSI) http://www.opensource.org/, 2005.

[7] LEON2 Processor, http://www.estec.esa.nl/wsmwww/core/soc.html http://www.gaisler.com/products/leon2/leon_down.html, 2005.

[8] Open Graphics http://wiki.duskglow.com/index.php/AboutOpenGraphics http://kerneltrap.org/node/5743

[9] E. Noulard & Y. Dufrenne, *The TSP whitepaper*, http://savannah.nongnu.org/projects/tsp, 2004.

[10] OMG adopted specification ptc/03-07-0, *Data Distribution Service for Real-Time Systems Specification (2003),* http://www.omg.org/cgi-bin/doc?ptc/2003-07-07

[11] M. Fleury, *Why I love Professional Open Source?*, 2003.

[12] *JBoss Group*, http://www.jboss.com, 2005.

[13] *MySQL AB*, http://www.mysql.com, 2005.

8. Glossary

COTS	*Commercial Off The Shelves*
AOCS	*Attitude and Orbital Control System*
AIT	*Assembly Integration and Test*
OSS	Open Source Software(s)
TSP	Transport Sample Protocol
TS2P	Testbed & Simulation Software Platform
GPL	(GNU) General Public License
LGPL	(GNU) Lesser General Public License
CORBA	Common Object Request Broker Architecture
HLA	High Level Architecture
DDS	Data Distribution Service