6

A Process Improvement Experiment through Process Modelling Technology

P. Coppola, P. Panaroni
Intecs Sistemi s.p.a.
Via Gereschi 32, - 56127 Pisa - Italy
Tel.: +39.50.545.111
Fax.: +39.50.545.200
e-mail: paolo@pisa.intecs.it

Abstract

Intecs Sistemi is an Italian software company deeply engaged in software developments for space systems. The growing role being assigned to software in the control of complex and/or critical functions of space system raised the need of a well defined and disciplined software production process to reach the required software quality.

This need is usually faced by the definition and regulation of the software process through Standards. However Standards have not solved the problem, and, to some extent, have even worsened it: thick documents, vague, ambiguous or incomplete are often hampering or slowing down production without actually assuring a better quality.

Process Modelling Technology is emerging as a set of models, techniques and tools to support effectively and efficiently a well defined and disciplined software production process through the notion of Process Centered Software Development Environments.

This paper reports a Process Improvement Experiment in a space on-board project through the adoption of Process Modelling Technology. The experiment is sponsored by the European Strategic System Initiative (ESSI). The tool Process Weaver, recently adopted by the DoD in its I-CASE initiative, was the selected process tool.

A formal BOOTSTRAP assessment was conducted at the beginning of the experiment, and repeated at the end, to reveal areas of improvement.

The paper will present and critically discuss these main topics:

1) industrial maturity of the Process Modelling Technology and corresponding support tools
2) ease of migration from traditional standards to process models
3) management and engineers acceptance/resistance to tools enforcing the process.
4) BOOTSTRAP assessments as a mechanism to monitor and measure improvement.

Keywords

Standard, quality, process modelling, process visualisation.

S. Bologna et al. (eds.), *Achieving Quality in Software*

1 OBJECTIVES OF THE EXPERIMENT

The experiment has been aimed at assessing application of process modelling technology to an industrial software development project by overcoming the limitation and troubles incurred by the use of traditional software development standards based on natural language.

The focus of the assessment has been on three main items:

1) maturity of the process modelling technology and corresponding support tools
2) ease of migration from traditional standards to process models
3) management and engineers acceptance/resistance to tools enforcing the process.

2 THE CASE STUDY PROJECT

The experiment has been based on a real industrial project. Identification of a suitable project was not an easy task and was constrained by various criteria:

- suitable for the technology to be experimented,
- feasible for transferability of results,
- in the right time frame,
- without affecting costs and schedule,
- carried out by motivated people.

2.1 Selection criteria

Suitable for the technology to be experimented

The project has been selected in order to be representative of a class of project for which Process Technology deployment is best suited. We then focused our search among the projects of our on-board software group where the applicable development standards are the more demanding and complex.

Transferability of results

We also looked at one project where adopted Standards were not excessively "unique" (ad hoc) to that project but, as far as possible, based-on/derived-from more generally applied standards. Since most space projects are now adopting the ESA PSS-05-0 Software Engineering Standard (ESA, 1992), a Process Model based on it will facilitated the transfer of experiment results to other project.

In the right time frame

It was not easy to match the ESSI experiment time frame with that of the on-going or planned projects but, fortunately, we could apply the experiment to a project from its start up to completion.

Without affecting costs and schedule

The funding from ESSI was rather effective to mitigate resistance on the extra costs induced by the experimentation, but a real hard resistance came from the perceived risks of project delays. Almost all software project are under tight schedule pressure, driven by external constraints. Application of a new technology is always perceived as a potential threat. Actually even good technologies (e.g. the Ada language) often provide tangible benefits only from the second project on. The use of Process Technology presented too many novel aspects (including organisational and psychological) and is of some concern to many managers.

Motivation of people
People acting as the object of an experiment, usually do not feel much "comfortable".

Fortunately we found a high motivation both from top management (that encouraged the on-board group to embark in the experiment) and project staff.

Moreover the project run in parallel of a company effort for ISO 9000 certification and Process Technology was clearly perceived as a fundamental step for achieving a well "defined" process.

2.2 The selected project

The selected project is a space software development of a critical on-board application. This class of applications is characterised by strong management requirements, thus matching the selection criteria of its appropriateness for deployment of Process Technology.

Furthermore, since the applicable development standards (covering the whole life cycle) were based on a slight "variation" of the ESA PSS-05-0 Software Engineering Standards, the modelling of the Process for this project was expected to be reusable also for other projects.

The selected project was not in a critical path and the risk of a little slippage was tolerated. However much care was continuously devoted to avoid any unnecessary "perturbation" that might cause a delay.

The application is the Input/Output Subsystem of the Fault Tolerant Computer Pool (FTCP) for a Guidance Navigation and Control (GNC) Avionics Test Bench. This pool of computer is composed of three replicated computers (based on SPARC and 1750A processors) linked by an Inter Processor Network and connected to several GNC busses based on the MIL-STD-1553B protocol.

It is sized about 10 KLOC of Ada code and involved an average of 4 persons for 15 months.

The project is carried out by Intecs Sistemi as principle software subcontractor of Matra Marconi Space, which, in turn, acts as prime contractor on behalf of the CNES (Centre Nationale D'Etudes Spatials). It is part of the Manned Spacecraft Technology Programme (MSTP), whose purpose it to prepare future European manned space missions.

While the hardware computers are triplicated (with a voting mechanism) the software is not diversified and the same version runs in parallel on the 3 computers in hot redundancy. Giving the safety requirements for a GNC system, the software is a highly critical components and in particular the Input/Output Sub-Systems is categorised as "safety critical" (class A software in ESA classification).

The major development methods in use have been the "Structured Analysis and Design Technique" (SADT) during the Requirement Analysis phase, the "Hierarchical Object Oriented" method (HOOD) during the Architectural and Detailed Design phases and Ada as implementation language.

The development environment included tools such as ASA (from Verilog), HOODNice (an Intecs Sistemi own commercialised CASE toolset), and two Ada Compilers: Verdix (native compiler) and TLD 1750A (cross compiler). The development environment is based on SUN SPARCStations (running Unix Sun OS).

3 THE EXPERIMENT

The project started on February 1994 for a duration of about 15 months (BEST-PM, 1995).

The major activities performed were centered around the following major areas:

1) rigorous modelling of the process
2) ProcessWeaver Tool
3) the 'Virtual Desk' metaphore
4) enacting the process on the case study
5) BOOTSTRAP assessment

3.1 Modelling the process

Our first concern was to define the Process Model with the major constraints of being compatible with the applicable standards and feasible for being implemented with Process Weaver. It was soon realised that the modelling power of ProcessWeaver presents some weaknesses. In fact the only available diagrammatic notation available to express process models is a tree of activities providing a plain Work Breakdown Structure (WBS).

We strongly missed a comprehensive high level diagrammatic notation capable to capture also data flow, activity dependencies, recursive and iterative activities, roles involved, etc. In addition this notation should have been easy enough for being understandable by Managers, Quality Engineers and in general all staff performing the process. Cap Gemini Innovation proposed an ad-hoc notation they have developed (not supported by the tool) (PROMESSE, 1995). Intecs Sistemi decided to adopt the notation resulting from the ESPRIT project SCALE (SCALE, 1993). The adopted process notation was an original improvement over SADT (Ross, 1977) (Ross, 1985) (Shepard, 1992) by adding concurrency, recursion, multiple perspectives and an implicit mechanism for backtracking to previous steps in the process (to model re-work).

The full life-cycle process has been rigorously modelled using this notation at a reasonable level of detail. The "model" was a good basis for clarification, improvements and training. Its rigorous and compressed form, made it much more effective than the thick narrative standards available.

This modelling effort represented a major achievement. An internal document, titled IMPROVE, reporting the full process description, was produced and is being maintained as the "master process". This maintenance can be seen as a kind of "continuous" process improvement. A consolidate version of the IMPROVE document will be proposed to become integral part of the company Quality Manual. This activity will parallel our transition from ISO 9000:1987 certification to ISO 9000:1984 certification that requires the revision/extension of some of our processes.

It is finally worth to mention that the adopted notation is independent from ProcessWeaver, therefore it could be used, in different contexts, in conjunction with other process enactment tools (e.g. Life*Flow, ProcessWise, etc.).

3.2 The tool Process Weaver

The tool Process Weaver, developed by Cap Gemini (Fernstrom, 1993), is supporting a rather novel technology compared to traditional CASE tools. More than a tool it can be perceived as an "environment" supporting "Process Centred Software Development".

The adoption of the tool had to be preceded by familiarisation of the staff involved on the basic principles of Process Modelling. The tool was quite easily installed and process toy examples were made running in a few hours. However full mastering of the tool required significant effort, including a formal course held by Cap Gemini experts.

3.3 The 'Virtual Desk' metaphor

The process modelling approach can be conveniently exposed with the so called 'Virtual Desk' metaphor. A metaphor is often an effective way to synthesise a complex new concept by exploiting its resemblance with a common and familiar concept.

The concept of Virtual Desk is very similar to that of Work Context used in the process modelling literature. However Desks can be more easily visualised and conceptually grasped.

A process is seen as the cooperative execution of several activities aimed at a common goal. Each activity is associated with one desk. Every time one activity is started a new desk is created exactly with the purpose of performing that activity. We may have a desk for the design, one for the coding of module A, one for the testing of module A, one for coding of

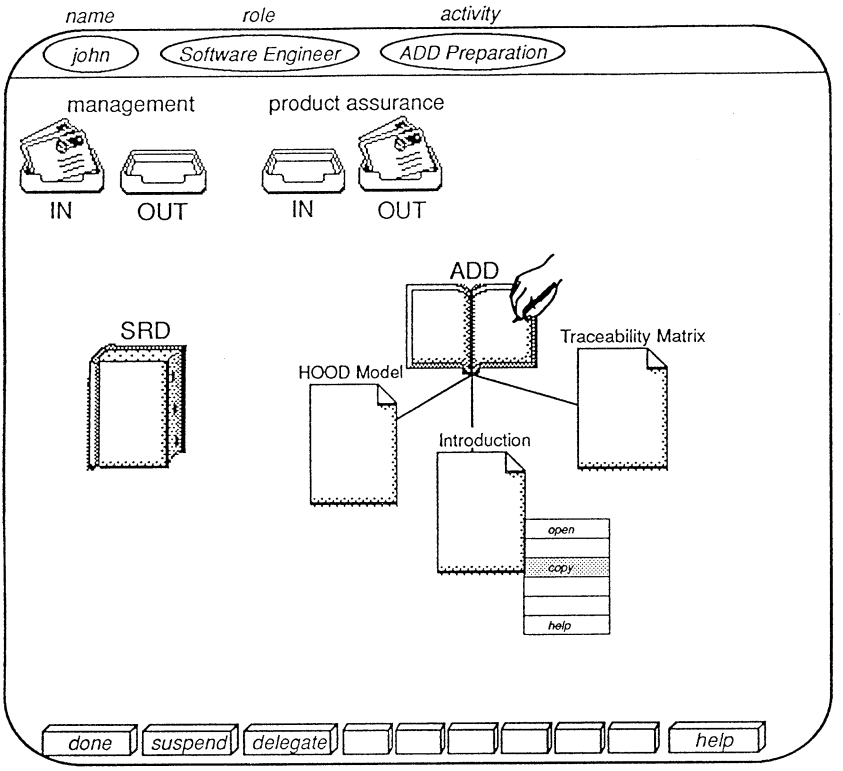

Figure 1 The Virtual Desk.

module B, etc. The desks for coding of module A and B will share the same type, but, are at all effects, different desks.

The whole process is then seen as a set of cooperating desks. A human can sit at a desk and perform part or all the activity, according to his/her role in the project and scheduled/authorized tasks. The same human can move over several desks or different humans can be assigned to the various desks. In principle, a human is responsible of the work performed while sitting at a desk.

An example of Virtual Desk is shown in Figure 1.

Virtual desks are not isolated "work islands" but cooperation among desks takes place through two basic mechanisms:

- exchanging of artefacts (document producer/consumer paradigm);
- exchanging of messages (conversation paradigm).

The distinction between artefacts and messages is rather fuzzy. One may claim that any artefacts can be exchanged by a message or viceversa that any message is a kind of artefact delivered to the other party. Common sense encourages to use the concept of artefacts for relevant and relatively stable pieces of information (e.g. artefacts usually are documents or source files managed under configuration control). At the opposite, messages should be used to model small and volatile pieces of information (e.g. a software problem report, a notification of completion, a request for delay, etc.).

The virtual desk supports these two paradigms of cooperation by:

1) automatically load on top of the desk all the input artefacts necessary to accomplish the activity. As soon as the activity has completed its output artefact/s is/are loaded on top of other desks according to the process rules. This implements the flow of artefacts from desk to desk (and the corresponding synchronisation).

   The desk "knows" what are the expected input artefacts, output artefacts and the allowed operators (e.g. tools) necessary to produce the output starting from the input.

   The desk supports the user by identifying all (and only those) artefacts and operators necessary to accomplish the activity. The user is not concerned about the selection of inputs, their completeness, their availability, selection of operators, applicability of operators to the input, etc. Everything is prepared on top of the virtual desk. The user mental load is reduced: all is creative energy is spent on the activity to be performed rather than on a quest for input and tools.

   Even more, the desk "knows" which operator is applicable to what artefact (e.g. a compiler on a source, a speller on a document, etc.) and even has knowledge to support proper sequencing of application of operators (e.g. first lint, only after compilation). The user is then warned against illegal sequencing of operators (e.g. first print, after spell) that might affect quality or productivity.

2) the exchange of messages between desks is supported by a strong paradigm well beyond and much powerful than the plain e-mail-like paradigm.

   Each desk has pre-defined a set of communication channels with other desks with which a communication can take place. Not every desk can communicate with any other desk: the process model defines a well predefined network of communication. A channel connect two desks. A protocol is associated with a channel such that communication on that channel shall stick to the defined protocol. Messages are named and typed. The respect of the protocol is controlled. The same message can even be broadcasted through many channels as long as it is compatible with the protocols of those channels.

An example of channel (e.g. status account) is one between the manager desk and the programmer of module A desk. Through that channel the manager may send a request for status report and the programmer shall respond with a status report.

The channel mechanism is able to discipline communication among the desks. If the manager is waiting for a status report the desk warns that such channel is pending waiting a response. Concurrent communications (conversations) can then occur between one desk and many others: each will have its own channel and its own "status of the communication".

Let's imagine a desk for a marketing staff member with a series of channels with each of the customers. New channels are dynamically created (on top of the desk) as new customers are identified. For each customer a communication is established (first contact, presentation letter, product brochure, offer, response, if positive answer, delivery, invoice, else make a discount, etc.). The human sitting on the marketing desk is supported by the desk that keep track of all open channels and their status. A new hired marketing person, "sitting" at that virtual desk, will easily take over the job, by proper continuation of message exchange always respecting the protocol imposed by the process model.

3.4 Enacting the process on the case study

Automatic enactment of the process by the staff of the on-going space project occurred incrementally according to the following steps:

Process tracking

The actual on-going process was manually tracked so that actual process events (e.g. start of an activity, review, re-work, etc.) could be used to validate the process being defined.

Implementation of the Virtual Desks

The various virtual desks needed to support the process were identified (more than 60) and designed. The design included:

a) definition the desk composition: input documents, output documents, tools, applicable standards, applicable plans, relevant reference documents, relevant reuse materials or good examples, etc.
b) definition of the desk layout (e.g. iconic representation of a source a test, etc.).
c) inter-desks relationships (which documents will flow from one desk to another).
d) definition of roles and staff enabled to work on the various types of desks (e.g. programmer, tester, QA, manager, etc.).

Process Weaver provided a nice user friendly and interactive interfaces to implement these virtual desks.

Model exercise

As soon as first fragments of the process model and corresponding virtual desks were available these were exercised by the project Quality Assurance Engineer to validate them against typical process patterns previously tracked.

Process Control

Once the whole process and all virtual desks were available, they have been used by the Quality Engineer only, to control and guide the process execution.

The QA engineer was actually "sitting" at the various virtual desks and used the information on the desk to "direct" the engineers in their daily activity in order to assurance conformance of the executed process with the model.

Manually the QA engineer gave (physically) the right documents at the right person at the right time and waited for the expected results. This manual activities was driven by the virtual desks.

In other words only the Quality Engineer was using the tool. The tool was then acting as an assistant to the Quality Assurance Role supporting the identification of what has to be done, with which dependencies, by whom, with which tools, on what input data, etc. Manually the Quality Engineer assured that the "real" process was enacted in accordance with the tool suggestions/prescriptions.

This phase of the experiment was called as "off-line" phase. It was useful to minimise interferences and perturbations on the on-going project while consolidating both the process model definition and it implementation through Process Weaver.

Model application

Then the experiment entered in its "on-line" phase. In "on-line" mode the tool directly interacts with the various roles of the process (manager, designer, programmers, tester, etc.) providing directly to them assistance on what has to be done. Thus the process was directly enacted by the tool without any "mediation" of the QA role: software engineers were working directly within the virtual desk.

Metrics collection
As long as the Process Model has been used, a tight monitoring of the enactment of the Process on the selected Case Study was performed. Monitoring included systematic metrics collection covering both process attributes (e.g. effort per activity, rate of re-work, etc.) and product attributes (e.g. size, complexity, quality of many artefacts including specifications, design, code, etc.).

3.5 Evaluating the results

In the context of the ESSI Application Experiment, the need for a "quantitative", further than "qualitative", evaluation and measurement of results was emphasized. In order to accomplish to this requirement, we decided to perform before the start of the experiment a BOOTSTRAP assessment and a "delta" assessment after its completion (BOOTSTRAP, 1993) (Kuvaje, 1995).

The formal assessment according to the BOOTSTRAP Scheme was conducted by 2I-Synspace (Friburg - Germany) .

The assessment evaluated at first Intecs Sistemi as a whole Software Producing Unit (SPU) by analysing standards, methods and practices at corporate level as evidenced by internal Quality Manual and associated procedures. The resulting score 3.0 placed Intecs Sistemi at the "defined" maturity level.

A second step of the assessment was the evaluation of two on-board space projects (similar to the one being under experimentation) to assess their project maturity. The resulting score was 2.8 for the best project and 2.0 for the worst, with an average of 2.4.

This was a clear indication that maturity of projects was lagging behind maturity of the SPU, and that an improvement was necessary to assure greater compliance of projects with applicable standards and norms.

The BOOTSTRAP assessment was not delivering just final scores (though important) but a full set of indications on weak areas and recommendations for improvement. All of them have been passed to the Quality Office as input for revision of internal Quality Manual and associated procedures.

The "delta" assessment, conducted after the completion of the experiment and centered on the FTCP project, resulted with a score of 2.6 . As far as the Intecs Sistemi as a SPU is concerned, we also experienced a small step forward at organization level (also due to the ISO 9000 certification obtained in the meanwhile). The obtained scores were under our initial expectation, even though, as long as the experiment was being carried on, we perceived that a significant improvement does not depend on the modelling of the process and its application on just one project but can be only consequence of a higher maturity level reached by both management, technical and QA staff. However, the "delta" assessment showed that we are on the right path.

It is worth to mention that, beside the measurement of process status and improvement, the BOOTSTRAP assessment was a useful exercise to better understand, with the support of external and independent experts, the company strengths and weaknesses. The wide set of technical areas covered was comparable or even superior to the SEI Assessment schema (for which we had a self-assessment experience). Coverage of ISO 9000 areas, for which there was some interest as the certification process was underway, was globally unsatisfactory: there were many ISO 9000 areas not well covered by BOOTSTRAP. We found very useful a score assigned to individual technical areas and a comparison with European averages.

4 LESSONS LEARNED

In general we can assert that the results of the experiment was positive, since it allowed us to better monitor and control the progress of each single task performed during the project

development, including both technical, management and QA ones. Nevertheless some points which need further improvement have arisen. In the following the major lessons learned during the experiment are presented.

4.1 The Process Model

The issue of Process Granularity is central: to which level of detail the process has to be decomposed and modelled? How many virtual desks are necessary?

While it is difficult to provide a general answer to this issue (Mittermeir, 1992), a pragmatic approach has been followed: the process has been decomposed up to terminal activities; a terminal activity, by norm, has few inputs, one single output and is performed by one single role using a small number of tools. When such a terminal activity revealed to be effort critical (i.e. it spend a significant percentage of the overall project effort), or quality critical (i.e. it generate a significant percentage of overall defects in the project) the activity has to be further refined.

The underlying philosophy is that a process model should be refined driven by the criticality (effort and quality) of the activities being modelled: there is no need for a very detailed breakdown of those activities which do not consume man/power and do not generate defects. On the other hand, critical activities, must be better understood in terms of their internal steps (sub-activities) and better controlled by rigorous enactment.

4.2 The tool

We found ProcessWeaver a useful tool for process "enactment", since it is easy to use and allows to efficiently track the ongoing activities. It is also worth to mention that it is satisfactorily reliable as we did not experienced serious problems during its usage. Nevertheless, at the time being, it is still naive for process "modelling".

A tool supporting the editing of process models using the selected diagrammatic notation was strongly missed. This lack was acknowledged by Cap Gemini that experienced the same problem in other similar projects (PROMESSE, 1995). This tool should also provide basic consistency analysis of the model being edited.

In addition it still lacks of some basic mechanisms to support advanced process modelling issues such as backtracking, early starting, etc.

4.3 The Process Enactment

By adopting the incremental approach presented above, a rather smooth introduction of this novel technology was possible.

In off-line mode the technology was only perceived by the QA Engineer and the first reaction from QA Staff was positive. They found in the tool a good and rigorous assistance for their day-by-day work of assuring that the process is executed as it is prescribed by the process model.

In on-line mode, when the tool directly and automatically drove the various roles in the execution of the process, people suffered of being too much constrained on what they had to do and when. In a typical process there are a lot of circumstances where activities are started, trying to met tight schedules, before they formally should be, in a sort of concurrent engineering (Frailey, 1993) (Petri, 1990). In addition the on-line mode was not capable to properly handle what has been called "process backtracking", that is the re-execution of the process from a previous activity (e.g. revision of specifications) in order to fix a problem or accommodate a change request (Suzuki, 1993).

On the other hand, there is a very positive reaction on the assistance given by the tool in terms of reducing the mental load (one task at a time), to identify the input and tools to be used and, last but not the least, synchronise activities between the various roles involved.

6 REFERENCES

BEST-PM, (1995) *BEST-PM - Beyond the Standards: Process Modelling*. ESSI Application Experiment nb. 10841, Final Report

BOOTSTRAP, (1995) Europe's Assessment Method. *IEEE Software*, May 1993

ESA (1992) *Software Engineering Standards*. PSS-05-0, issue 2,

Fernstrom C., (1993) Adding Process Support to Unix. IEEE Software, September 1993

Frailey D. (1993) Concurrent Engineering and the Software Process. in *Proceedings of the 2nd International Conference on Software Process*, Berlin,

Kuvaje P., (1995) BOOTSTRAP: a Software Process Assessment and Improvement Methodology. in *Objective Quality Software Symposium,* Florence, May 1995

Mittermeir R.T. et al. (1992) Stepwise Improvement of the Software Process in a Multidimensional Framework, *Annual Review of Automatic Programming*, **vol. 16**

Petri J., Pulli et al. (1990) Concurrent Engineering for Real Time Systems, *IEEE Software*, November 1993

PROMESSE, (1995) *Final Report*. ESA contract nb. 10081/92

Ross D.T., (1977) Structured Analysis (SA): A Language for Communicating Ideas. *IEEE Transaction on Software Engineering*, January 1977.

Ross D.T., (1985) Applications and Extensions of SADT. *COMPUTER*, April 1985

SCALE, (1993) *Process Modeling Formalism Definition*, ESPRIT project nb. 6334

Shepard T. et al. (1992) A Visual Software Process Language, *Communication of the ACM*, April 1992

Suzuki M. et al. (1993) A Formal Method of Re-execution in Software Process, in *Proceedings of the 2nd International Conference on Software Process*, Berlin, 1993

BIOGRAPHY

Paolo Coppola

Paolo Coppola is the head of the Special Systems Section of INTECS Sistemi. He is responsible of the development of critical software systems, in particular for spacecraft on-board applications.

Paolo Panaroni

Paolo Panaroni is the head of the QA & Methodologies Section of INTECS Sistemi. He is responsible for the overall Company Quality Management and of internal Quality Manuals in compliance of ESA, NATO and ISO software standards. He is currently President of the Ada-Italy Association.