# Impact of software review and inspection

I. Aleksandrov, V. Amaral, A. Amorim, E. Badescu, D. Burckhart, M. Caprini, L. Cohen, P.Y. Duval, R. Hart, R. Jones, et al.

# Impact of Software Review and Inspection

*I. Alexandrov[1], V. Amaral[2], A. Amorim[2], E. Badescu[3], D. Burckhart[4], M. Caprini[4,9], L. Cohen[5], P-Y. Duval[5], R. Hart[6], R. Jones[4], A. Kazarov[7], S. Kolos[4,10], V. Kotov[1], D. Laugier[5], L. Mapelli[4,11], L. Moneta[8], Z. Qian[5], C. Ribeiro[2], V. Roumiantsev[1], Y. Ryabov[7], D. Schweiger[4], I. Soloviev[4,10]*

1 Joint Institute for Nuclear Research, Dubna, Russia
2 Lisbon Institute of Physics, Lisbon, Portugal
3 Institute of Atomic Physics, Bucharest, Romania
4 CERN, Geneva, Switzerland
5 Centre de Physique des Particules de Marseille, IN2P3, France
6 NIKHEF, Amsterdam, Netherlands
7 Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg, Russia
8 Section de Physique, Universite de Geneve, Geneva, Switzerland
9 On leave from 3.
10 On leave from 8.
11 Spokeperson for DAQ-1 project

**Abstract**

The Software Review has been part of the software process for the ATLAS Data Acquisition Project DAQ/EF-1 since its start in 1996. Taking the form of an informal review, the software is presented at each of its phases to the group during a meeting and then discussed aiming for an accept/reject decision.

In 1998 the more formal Software Inspection process based on Tom Gilb's method has been introduced. Software Inspection is a quality improvement process of written material including code with the objectives of defect detection and subsequently defect prevention. Improvements are achieved for the end-product as well as for the process of document and code production. Flexibility built into the working process allows methods and rules to be updated in response to participants' suggestion and change in technology. It provides ongoing integration and education of participants.

A number of Software Inspections on Requirements, Design, Code and its documentation have been performed. A project specific database of metrics has been established since the introduction of such inspections. Results show the importance of software inspection for the entire software lifecycle and in particular of the requirements to ensure defect detection early in the production phase. Areas for improvement have been identified in the fields of coding conventions, naming rules and guidelines for establishing and using document templates in order to facilitate communication and integration of individual project components.

Keywords: Quality Assurance, Software Development Process, Inspection, Review, DAQ, Atlas,

## 1 Introduction

This paper reports on the evaluation of review and software inspection as part of the software development process (SDP) in the context of quality assurance for the ATLAS [1] DAQ Back-end software sub-system [2][3][4] which is part of the DAQ/Event Filter Prototype "-1" project for the ATLAS experiment.

## 2 The Atlas DAQ Back-end Software Development

The Back-end software encompasses all the software to do with configuring, controlling and monitoring the DAQ but specifically excludes the management, processing or transportation of physics data. This project has been divided into 12 components[2][3]. Typically, a single institute has taken responsibility for developing a component thereby simplifying communication and reducing travel. The same individuals tend to follow a single component through the various phases.

By applying a **Software Development Process** the development has been divided into a number of sequential phases intended to help pace and organise the work. Each phase has been defined to produce an obvious deliverable, i.e. document and/or code. Each deliverable from each phase is **reviewed** before progressing to the next phase. The phases are: collect requirements; identify and evaluate candidate technologies and techniques capable of addressing the common issues identified from the requirements; produce a design for each component covering the most important aspects; refine the design to add more detail; implement and unit test according to the design; integrate with other components. Later more **formal inspections** were introduced using a system of peer review supported by guidelines and checklists for documentation and code.

## 3 Informal Reviews

Reviews have been performed since the beginning of the project. They take the form of presentations followed by discussions during open meetings with all developers involved in the project. The developer of a component from a participating institute prepares the document. Then one or more colleagues, often the project leader, go through it and comment on it. When corrections have been made the document is made available to the project members who are supposed to read it. Reviews are organized during the monthly Back-end meetings where a developer presents the status and results from a development phase or a list of items to be investigated for the next phase. The aim is to inform other project members and also to receive feedback. Since the advances of the project software is governed by regular releases, the list of additions and changes per component for the next release is also presented. During the subsequent discussion open items are clarified, the relation to other project components is discussed and the component in its phase is accepted or suggestions for modifications and enhancements are made. Decisions for further development are taken.

### 3.1 Results

Each component of the Back-end project has been reviewed before progressing to the next phase. Code has generally not been reviewed. In some cases code fragments were identified to be sufficiently interesting or informative and have been presented as well.

The review of each deliverable in the Back-end project at each of its phases has lead to a coherent set of end-product components. Modifications of a component due to the evolution in ideas or due to technical constraints could be accommodated in agreement with other components and their developers. In one case the discussion has even lead to the development of a new component, the Information Service [2][3].

Reviews have created the basic culture for individual developers located in geographically distinct institutes to form a team and to work successfully in a common project. Presentation of their work in the review meetings showed that their peers valued their work, kept them informed about the work of their colleagues and discussions during and outside the meetings increased communication. A drawback was that the reviews on the documentation and code were not performed thoroughly and often team members did not find the time to read the documents before the meeting. This has led to the introduction of more formal inspections with its structured organisation and clear allocation of working time for inspection work.

## 4 Software Inspection

Software Inspection is a technique for achieving quality control for written material and for identifying associated process improvements. Its regular application and success in the software industry suggested its introduction to the on-line computing environment. It has first been invented by M. Fagan at IBM in 1976 and has since been improved [5] and adapted by many major software companies. It is part of quality assurance in the SDP, it complements automatic checking tools and is performed by real people.

### 4.1 Principal Aims of Software Inspection

Inspection as part of the **Defect Detection** Process is performed before testing, and to complement testing. The relation to the SDP is illustrated in figure [1]. Documents are checked for cleanness and consistency against rules. The objective is to identify and correct major defects in the candidate product before releasing it from the current development phase.

The **Defect Prevention** Process is concerned with learning from the defects found, and suggesting ways of improving the processes to prevent them from re-occurring in the future. It involves process analysis which is carried out off-line from the normal inspection of specific documents. Team participants profit from the experience made during the inspection when producing their own work while the inspection process is improved on participant's suggestions and according to changes in technology.

**On the job training** is a valuable benefit of a dynamic and open inspection process. It provides implicit integration and education of people which are new to the project. It helps in building up the project's working culture. A basic set of process guidelines, checklists and rules is available for convenient entry into the project while being open to easy modifications and additions of new ideas.

### 4.2 The Software Inspection Procedure and the Inspection Team

The Back-end inspection process [6] applied in the Atlas Back-end software subsystem is based on the inspection method developed by Tom Gilb [5][7]. Inspection is based on a well defined and organized procedure following a sequence of actions including meetings and work done by individuals. It is important to provide this basic framework in order to avoid inspection team participants wasting valuable time. Deviations from the proposed structure are allowed whenever it increases overall efficiency.

Inspection in a project must be introduced and managed by a dedicated person, the **Inspection Manager**. He has the overall responsibility to introduce the inspection process to the project and adapt it to the environment. In collaboration with the inspection leaders and interested project members he proposes the inspections to be performed, keeps track of changes to the process, procedures, rules and guidelines in order to keep the Inspection Process dynamic and up to date. The inspection manager must have the backup of the project leader yet should be a different person.

Each Inspection is organized by an **inspection leader**. He assembles the **Inspection team**, which consists of the **document author**, three to five **inspectors** and the inspection leader. Experienced peers and one or two novices are invited to each inspection. This is particularly appropriate when they are likely to be involved in the production of a package which depends on the product. Inspectors who are involved more directly in the project are more likely to get the correct and important feeling that their work is valued and useful to the project. In this context it is equally important that they can see that their suggestions are followed up. The inspection leader has to plan and lead the individual inspection process, choose the inspectors, plan and moderate meetings and make sure rules and procedures are followed and updated if necessary. He also has to make sure Inspection team members understand the purpose, benefit and procedure of the Inspection process, collect process improvement suggestions and follow them up and collect basic metrics.

The Inspection Process as displayed in figure [1] is managed by the inspection leader. The author informs the inspection leader that his document is ready to be inspected. In case of implementation inspection the code must have passed automatic testing tools like checks for memory leaks.

At the *Planning & Entry* the inspection leader checks that the document meets the entry criteria for Inspection (quick quality check) and appoints 3-5 peers as inspectors. He prepares the lists of documentation, rules and standards. He may give individual instructions for checking to peers.

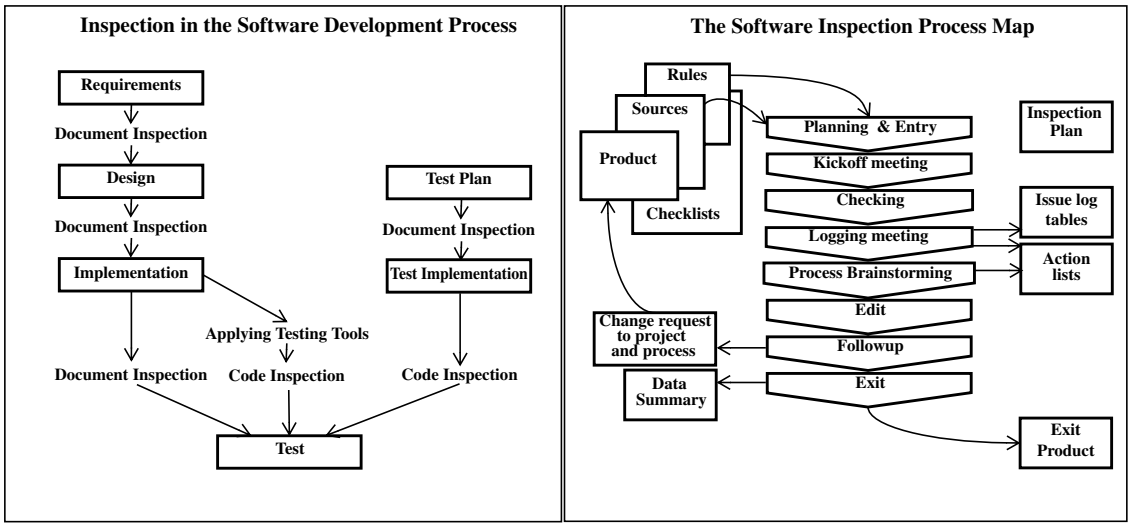


**Figure 1: The the Software Development Process and the Software Inspection Process**

Then he organizes the *Kickoff meeting* (15mins), where he gives time scales and other instructions to the peers and the author of the document explains in general terms the structure of the document. *Checking* is performed by each peer individually, who logs each defect in a table.
Ideally all documents in the software development process are inspected. Three development phases are concerned with their corresponding documents which are: ***Requirements*** specification, Architecture ***Design*** together with the Test Plan, and Detailed Design together with ***Implementation*** (code) and Users Guide. Documents should only be accepted for inspection, when its source document has been inspected and accepted.
Documents are compared against mother documents, rules and standards, aided by checklists, and checked for cleanness and internal consistency. Comments and footnotes are not inspected. It is recommended to concentrate on the finding of major defects. A defect is defined as a violation of standards, official rules, in-house rules, and any mismatch as compared to the mother document or to internal consistency. A major defect would propagate to the end product and could possibly cause problems at testing time.
When the peers have finished checking at the agreed date the inspection leader organizes the *Logging meeting* (max. 2 hours) where major defects are discussed and their acceptance or rejection is recorded in the inspection issue log.
The *Brainstorming meeting* (5-30 mins) follows shortly after the logging meeting. In this meeting feedback on the inspection procedure and change requests to the rules and standards and to the inspection process itself are handled. Necessary changes are discussed and recorded by the inspection leader. He also ensures that the necessary actions are performed. The document author is expected to undertake issue analysis and correction action. He *Edits* the document accordingly or rejects the item on the list by adding an explanatory comment. The inspection leader *Follows up* the change requests issued at the brainstorming meeting and keeps in contact with the author.
The product is ready for *Exit* from inspection when all the items on the author advice log have been satisfactorily worked on or been rejected, and when the data summary and the metrics on the inspection has been delivered and has been agreed by author and inspection leader.

Regularly updated concise **checklists** are provided for the inspection team members to help in their efficient participation. The development of **in-house standards** which are available for requirements, design and code allows to separate commonly agreed **rules** from personal taste. Checklists as well as standards and rules are part of the framework, they are frozen during the lifetime of one inspection. Regular updates help to ensure an up-to-date working basis. Valid versions are available on the project Web pages [6].

An easy to use FrameMaker **template** is provided for the issue log table of each inspection and is used by the peers. Inspection results, the filled and edited issue log tables, minutes and action lists are stored under afs with access groups set up individually for each inspection. **Access** is **restricted** and granted to team members of an inspection only. This protection via acl lists has been introduced to avoid misuse from people who may have no knowledge of the purpose of inspection. However each team member is free to pass the information with the appropriate comments to his colleagues.

### 4.3 Inspections performed in the Atlas DAQ/EF-1 Back-end sub-project

More formal inspection as described above was introduced as an evaluation while development was already well under way. Therefore only a limited number of deliverables which were produced could be inspected, only two of them in more than one phase. Three inspections on requirements, two on design and four on implementation documents were performed on the components described in [3].

Requirements Document inspection included the Back-end software Diagnostics System (DS), the Integrated Graphical User Interface (IGUI) and the Data Access Library (DAL). Design Document inspection included the Test Manager (TM) and the Back-end software Diagnostics System design (DS). Code inspection included the Inter process Communication Package (IPC), the Message Reporting System (MRS), the Information Service (IS) and the Data Access library (DAL). In total 8000 lines of code which is about 20% of the in-house written code of the Back-end software and 180 pages of documentation were inspected. 19 inspectors and one inspection leader participated in the inspections.

### 4.4 Inspection Results

The evaluation was performed during the phase when software inspection was introduced to the project and it team. Initially a steep learning curve was observed, because inspection was started basically from an initial level of zero. A high number of issues of varying severity was found. With the implicit education in the team the number of issues found became lower while concentrating more around major defects. Authors who have participated in software inspection previously as an inspector generally provide a cleaner deliverable from the start.
The **deliverables** of each inspection were:

- The **corrected documents and code**.
- One **issue log table** per inspector, filled by him while inspecting and before the logging meeting. Non-trivial issues were discussed during the meeting. Subsequently the tables were edited by the author and cross-checked by the peer. The number of issue logs found depended on the type of inspection (requirements, design, code), the project phase, the entry conditions (availability of a clean set of rules and guidelines), the experience of the inspectors and on the counting system. Variations between 10 and 200 issue logs per inspection were found, minor issues included. Issue counting in the future will be restricted to major issues.
- **Change requests** and **action lists** were discussed and recorded during the meeting. During the inspection and also during the logging meeting the need for changes of checklists or rules came up. Certain rules turned out to be too restrictive or not sufficiently concise, checklist had to be updated according to the evolution of the project and improvements concerning the inspection process and the development process itself were suggested. For example, for the requirements inspection a recommendation for the use of the words ‘shall’, ‘should’ and ‘may’ was found to be necessary; for coding the need for the cmdline package for command line parameters was discovered and exit status conventions were suggested. Action items came up for issues which were beyond the scope of the particular inspection and had to be handled outside, for example feasibility tests. The inspection leader followed the change requests and action lists up.

**Requirements inspection** were found to be the most important. They are the first in the chain. Requirement inspection is done because according to experience [7] it takes about one hour to find a major defect by inspection at an early stage and about nine hours when testing. A defect may propagate to the end product even with perfect code. Requirement inspection are also the least time consuming because no or few mother documents must be read and they are generally a few pages long only. They turned out to be useful to re-visit and clarify strategy and goals. In one case we experienced the need for a redefinition of the component composition - a component was split into two distinct parts.

**Design inspection** are the hardest to perform. It was found to be difficult to define a good set of guidelines which is not trivial but also not too restrictive. The use of design patterns will be investigated.

**Code inspection** are the most time consuming ones. The number of documents involved is high: code must be checked for internal consistency and against coding rules, the users guide and the implementation documentation must be inspected and compared against the design and requirements documents. Automatic checking tools were employed where possible. Sampling has been performed in some cases and will be employed regularly in the future while concentrating on critical areas.

## 5 Experience

A guiding principle for the introduction and for the use of the software inspection method is that *everything is allowed which helps improving* the product, the overall production process including communication amongst project participants or the inspection process itself *while keeping consistency and improving efficiency.*

In the HEP environment a number of factors must be considered being different from the industrial world and their working methods and therefore software inspection has been adapted according to our needs. Participants in a project in the HEP DAQ environment are each expected to fulfil all the roles required for the development of a product in a sequential or parallel order whilst industry is usually structured into a development department, a quality assurance department, a testing department, etc. Each of those departments has the expertise for the particular project phase or task.

In order to reach the same level of expertise project members with varying professional background would have to be trained formally in each of the areas and have working experience for a significant amount of time. In the case of the Back-end project one inspection leader was trained in the inspection method but no formal training of inspection team members could be done because of constraints in cost, time and geographical distribution of project participants. The group is comprised of a few Cern resident people and a number of collaborators from outside institutes with additional duties. For the same reason inspection could not be imposed by a project management as this is common in industry. Participants were convinced of the benefits by receiving personally the appropriate explanations and by making positive experience.

### 5.1 Evaluation results

The well defined component structure of the Back-end project and the division of the development into phases with obvious deliverables by the SDP provided the necessary organisational basis for the successful introduction of software inspection. Inspection itself supports the SDP by ensuring good quality deliverables backed up by the establishment of standards and rules. Particular experience has been gained concerning the following points.

Introducing the inspection method gently when informal reviews are already in place reduces resistance. Informal reviews have provided the basic culture to make the subsequent introduction of formal inspection possible. It seemed necessary to provide a more formal framework in order to pinpoint reviewers to their work and to give them the necessary justification for spending valuable working time.

Specially in the introductory phase it is important to insist on real logging meetings as opposed to pure electronically communicated inspection. This is highly appreciated for integration and educational benefit in particular by new project members. For efficiency logging tables are filled during the inspection work and the author has a quick look at them before the meeting accepting the obvious issues. This allows everyone to concentrate at the meeting on the critical points. Experienced peers can take part via phone. In evaluation phase the outcome expressed as a number of majors found per page could not be used as a comparable result as each inspection had its particular circumstances. Success could mainly be shown in the clear acceptance and appreciation of software inspection by the project members and in the improved end product.

The inspection leader must take care that inspectors are not overloaded by the work for inspection and choose peers who are interested in the candidate product. Everybody has his load of work, schedule and responsibilities in their home institutes. Careful planning is vital. He must provide the basis and must make it possible for potential peers to take part in an inspection. He must introduce the idea of constructive criticism into the team. It must be emphasized that nobody is expected to be perfect and also inspection will not find every defect. Perfection is to not necessary, not efficient and therefore too expensive.

If participants can take an active part in the construction of the working framework by building up rules and guidelines and achieving agreement on working methods they are likely to accept it with ease.

Creativity of inspectors in particular in ways of finding hidden majors must be encouraged to increase productivity. Their contribution is and must be recognized of being important.

For increased efficiency inspection work can be split amongst peers for example: check against coding rules; consistency with kin documents like Users Guide (check API); implementation note; consistency with design document; looking for special areas in code like loops, etc.; functional division: i.e. in a server/client application one inspector may look at the server, another at the receiver or sender part.

Initially inspectors found many ‘minor’ issues. This improved with time and experience because authors who also have been inspectors brought their acquired knowledge into the new document and code. Checkers could then concentrate on the more serious problems.

Software inspection has two fundamentally different aspects:

- It is on one hand an exact, **dry, formal**, and technical task. Because of its time-consuming nature ***efficiency*** (effectiveness within time) is important to avoid boredom and wasting inspectors time. A solid process framework with clear instructions, entry condition checking, focussed meetings, project specific checklists and rules help save time. This must be balanced with the ***flexibility*** of handling each inspection individually while recording deviations from the standard. Build-in change management concerning rules and the process itself help in keeping inspection lively and up-to-date.
- Software inspection **deals with people**. In the HEP DAQ environment people with many different technical, educational and cultural backgrounds come together and fear to be judged by foreign colleagues. Open criticism is barely supported. Already the name ‘inspection’ may provoke negative feelings and psychological aspects have to be taken seriously when introducing the inspection method to a project. Explanations to the individual emphasizing the aim - the improved end-product as a result of team work as opposed to criticism of the person - and privacy of direct inspection results within the inspection team are vital. However having made the active and then passive experience of constructive criticism and overcoming those fears helps people to integrate, in building up trust amongst colleagues, even if remote, and in constructing a common working culture. The vulnerable position of an author in the inspection process must not be neglected. Peers must demonstrate helpfulness.

### 5.2 Conclusions and Future

Reviews have proven to help stabilize the Software Development Process and are an integral part. People get the chance to present their work, receive feedback and integrate it better into the common project. Reviews will remain an integral part of the SDP.

During the evaluation of software inspection the method has been successfully introduced to the project. Project members became familiar with the method, the procedure was adapted to the project and the working environment and sets of rules and checklist were established. Authors appreciated the gain in quality of their work. A second more stable phase can be envisaged now. The aim is to apply inspection to all the deliverables in the project in the three basic development phases of requirements, design and implementation in particular at the start of the life cycle, which fits well to the start of the development of the final Atlas DAQ software. Training of inspection leaders will be required. ‘

‘Light’ inspection with the same checking criteria but through sampling, by concentrating on critical areas and with faster turn around time should be used in more cases, in particular for evolutionary delivery development. The systematic use of Code checking tools [8] will help reduce the workload on inspectors. Open logging meetings as opposed to electronically conducted inspections will be kept as they have proved to be valuable in keeping a common working spirit. Dynamism and flexibility for the process must be kept alive and rules and checklists continuously be improved according to the ‘majors’ found and to changes in technology and the evolution of the project. The quality level can be raised smoothly by raising the level of standards and rules.

## 6 Acknowledgments

We would like to thank the document authors of the Back-end project for having accepted their work to be inspected. We are grateful to all the peers for having spent their time improving their colleagues’ product and for keeping a positive style of encouraging and constructive criticism. The inspection manager wants to acknowledge specially the valuable inspection team leaders course by Tom & Kai Gilb, followed by personal consultancy, which helped understand the spirit of successful software inspection and avoid common mistakes when introducing and conducting inspection in our project.

## 7 References

1 ATLAS Technical Proposal, CERN/LHCC/94-43 (ISBN 92-9083-067-0)
2 I.Alexandrov et al., “Back-end sub-system of the ATLAS DAQ prototype”, CHEP-98, Chicago
3 D. Burckhart et al., “Back-end Summary Document”, ATLAS internal note ATL-DAQ-2000-001
4 I.Alexandrov et al., “The Performance and Scalability of the back-end DAQ sub-system”, CHEP-2000, Padova
5 Tom Gilb, Dorothy Graham, “Software Inspection”, Addison Wesley Longman, Inc.,1993
6 “Software Inspection in the ATLAS DAQ/EF -1 project”,http://atddoc.cern.ch/Atlas/DaqSoft/sde/inspect/Welcome.html
7 Kai & Tom Gilb, “Inspection Team Leader Training Course”, http://www.result-planning.com/
8 E. A.Ribera et al, “An evaluation of tools for the static checking of C++ code”,CHEP-2000, Padova