Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT

Information and Software Technology 45 (2003) 671–680

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

# Efficient management of inspections in software development projects

A. Chatzigeorgiou[a,*], G. Antoniadis[b]

[a]*Department of Applied Informatics, University of Macedonia,156 Egnatia Str., Thessaloniki 54006, Greece*
[b]*INTRACOM S.A., Thessaloniki, Greece*

Received 28 October 2002; revised 26 February 2003; accepted 4 April 2003

**Abstract**

During the last two decades a universal agreement has been established on the fact that software inspections play a fundamental role in improving software quality. The number of software organizations that have incorporated formal reviews in their development process is constantly increasing and the belief that efficient inspections can not only detect defects but also reduce cycle time and lower costs is spreading. However, despite the importance of the inspections in a software development project, scheduling of inspections has not been given the necessary attention so far. As a result, inspections tend to accumulate towards internal project deadlines, possibly leading to excess overtime costs, quality degradation and difficulties in meeting milestones. In this paper, data from a major telecommunications software project is analyzed in an effort to illustrate the problems that can arise from inefficient planning of inspections and their related activities.


*Keywords:* Software engineering; Inspections; Planning; Software quality; Software project management

## 1. Introduction

Since the first published description of the inspection process by Michael Fagan [4] software inspections are gaining increased acceptance by both software developers and project managers. The reasons for this increasing practice of inspections, which, however, has not yet led to widespread usage according to the Software Engineering Institute [10], are not limited to their usefulness in improving quality by preventing defect leakage from one life cycle activity to the other. Industry experience shows also that through inspections, cycle time is reduced, costs are lowered, process visibility is increased and also programmers' capability is improved [4,6,20].

Inspections form an integral part in all phases of software development and apart from code reviews, cover all software artifacts, such as requirements, specifications, architectures, design, and test plans [11,12]. Although inspection details vary according to the employed methodology and the target project, a set of common elements can be identified. These include a number of well-defined inspections steps (e.g. preparation, meeting, rework), well-defined inspection roles (e.g. moderator, author, inspector), formal collection of inspection data and a supporting infrastructure [1].

In almost every study dealing with inspections [1,2,10,20] planning of inspections is identified as the initial stage of the inspection process. However, the purpose of planning in the above sense is to define the goals, the objectives and the methodology of the inspection [6] rather than to coordinate the inspection process with other project activities. Even though the importance of scheduling inspections in time has been addressed in Ref. [7] and [22], the arrangement of staff, people and time for the inspection is performed in short term before the inspection and definitely not earlier than the time when the item for the inspection has been prepared and checked to see whether it meets certain entry criteria [4–6]. In other words, the planning of the inspections is not performed well ahead the project initiation as it is done for all other project activities and it is not related to the project master plan.

Approaching inspection activities only in a procedural manner and not by taking into consideration project management requirements like time scheduling and personnel capacity issues, can give rise to several problems, especially in large software development projects. These troubles can vary from issues affecting only the inspection

* Corresponding author. Tel.: +30-2310891886; fax: +30-2310891875.
*E-mail addresses:*achat@uom.gr (A. Chatzigeorgiou); gant@intracom.gr (G. Antoniadis).

0950-5849/03/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0950-5849(03)00074-0

efficiency and the product quality, to others that put into danger the complete project. Risks that can emerge when inspections and resulting rework are not properly scheduled, have also been mentioned by Fagan in Ref. [4]. Some of these potentially inspection-related problems are listed below:

- Difficulties in meeting milestones due to unavailability of human resources (especially experts) for performing the inspections or due to bottlenecks in approval bodies and committees.
- Product quality degradation due to reduced inspection and preparation time and the subsequent reduced number of defects being discovered during the inspection.
- Cost increase due to excess overtime, which have not been initially planned and due to higher costs for fixing faults in later development stages.

The literature review reveals that although many previous works emphasize some of the problems caused by insufficient inspection and preparation time [2,9,21], or by delays and costs introduced from a large number of inspection meetings [12,13,15], they do not relate these issues with inefficient planning of the inspections. Software project management textbooks emphasize both the importance of careful planning activities and the use of software inspections as a verification and validation tool. However, reference to the risks related to inefficient inspection planning is limited [7,8,22].

In this paper, by performing post-mortem analysis on a large-scale telecommunications project, we attempt to illustrate inspection process complications that can be avoided by efficient scheduling of inspections. The findings will be highlighted using a number of indicators. Further analysis attempts to shed light to the reasons behind the identified weaknesses and to establish general guidelines for improving software project management concerning inspections.

The rest of the paper is organized as follows: in Section 2, information on the case study is given including the specific project characteristics and applied management methodology. The research questions, the collected measures and the possible threats to the investigation are enumerated in Section 3, while Section 4 summarizes our observations. A discussion on further reasons that complicate the planning of inspections is made in Section 5 while in Section 6 a set of general guidelines for improving inspection scheduling is proposed. Finally, we conclude in Section 7.

## 2. Case study

To reveal some of the side-effects that inherently reside in the application of the software inspection process in today's software development practices, data from a major telecommunications project is analyzed. In this section, characteristics of the project that has been selected are reported, including the project planning techniques that are employed by the organization and the review process that is applied.

### 2.1. Project characteristics

The scope of the project was to add new functionality to a working application software package by performing the necessary modifications in the appropriate software components (as component is considered any discrete executable part of the software application, accompanied with the necessary documentation and having a well defined functionality in the system). The 'Detailed Design', 'Coding' and 'Unit Testing' activities of the project have been studied in a period covering 12 months. A total of 25 engineers participated to these phases, most of them by being fully allocated to the project. Another seven engineers participated only as inspectors. The number of the impacted software components was 45 ranging from 1 to 27 KLOC, giving a total of 283 KLOC. The modification grades of the components ranged from 1 to 11%. In total, 97 man-months were spent to the design and inspection activities under study.

### 2.2. Project planning

Since directives and limitations set at the project planning phase are crucial for the distribution of several activities throughout the course of the project, it is important to outline the main steps that have been followed for the project set-up and steering. The methods that have been used are based on common principles of software production operations management [23]. As a prerequisite for the project initiation and due to work performed in earlier design phases, a stable input existed concerning the functionality, which would be implemented and the approximate estimation of the required effort per component. Additional effort was also foreseen for overheads related to other technical or administrative activities (fixing known faults, inspections, reporting, meetings). Taking into consideration the required design effort and the available human capacity, the project execution period was agreed with the line organization and the required human resources were reserved for the project. In order to use the personnel experience in the most beneficial way, engineers familiar with specific components were allocated to design work related with these components. In cases in which the engineer had little or no experience of the assigned component, additional time was given to balance the designer's lower productivity. On the contrary, in cases in which the engineer was very familiar with the component or had a good knowledge of the impacts, the planned time was shorter. Normally, the detailed design, coding and the unit

testing with all related artifacts of each component were assigned to the same person. Exceptionally, in cases of significant modifications, more than one engineer was allocated to one component. In cases of small impacts to specific components, partially allocated engineers with experience to the certain component were used. All engineers were asked to check and commit their allocations. Therefore, what was fixed at the end of the resource allocation procedure was the working period for each individual, his/her average allocation percentage over this period and his/her responsibilities. What was variable or unknown was the deviation of the real effort per component from the initial estimation, the distribution of the design effort over time, any unpredictable leaves of absence, any calls for participation to other activities like inspections, meetings, etc. Within this framework, each engineer had the flexibility to define the execution sequence for his/her own tasks, to share his/her time between parallel responsibilities and generally to configure his/her personal plan. Overtime around 10–15% has been considered acceptable in peak periods. The purpose of the allocation plan was to distribute smoothly the design work between the personnel and to avoid idle periods or work peaks for the department staff. Allocation plans were reconsidered at the beginning of each design increment, at personnel resignation and at any time the project deadlines were in danger.

### 2.3. Inspection process

The inspection process was based on the well-established methodology outlined by Gilb [6] and all participants in the project had received formal training concerning the used methodology and infrastructure. Detailed description of the application of Gilb's inspection methodology to the particular sort of projects can be found in Ref. [19].

The inspection procedure was initiated by the 'quality coordinator' of the project who, based on the design deadlines and the engineer's personal plans, was in charge of setting the inspections teams and scheduling the dates of the inspection meetings. The inspection team consisted of the author of the work artifact (document or code) and two or three inspectors. In a few exceptions only one inspector participated. One of the inspectors acted also as the moderator of the inspection. The inspectors had some kind of relation with the inspected item: they were either engineers who wrote the specifications of the introduced functions or they were experts on the specific software component or they were project members introducing similar or complementary modifications in a different software component.

According to directives given by the project, the inspected work artifacts should had been ready and 'frozen' 1 week before the inspection meeting. For this purpose the author had to store the corresponding documents in electronic libraries on the planned dates. Individual preparation and study of the inspected item were performed from this point until the inspection meeting. Any findings during the inspection meeting (defects or improvements) were recorded in an 'Inspection Record'. After the inspection, the author implemented the recommended rework and stored again the document in the same library with a stepped revision. The inspection record was also stored in the library. The whole procedure was monitored and coordinated by the inspection moderator.

## 3. Post-mortem analysis

This section states the main research questions to be investigated, enumerates the measures that have been collected for performing post-mortem analysis on the selected case study and discusses the potential threats to the validity of the analysis.

The overall argument of this study is that software inspections due to their role in validating design steps within a software development project tend to accumulate at specific periods during the course of the project. Therefore, if they are not scheduled in a way that takes into account the required resources, time and effort, and if not coordinated with other project activities, they will cause peaks and bottlenecks in the use of project resources with negative consequences on the cost and the quality. Consequently, the main question that is investigated is whether inspection effort accumulates in specific periods, irrespectively of the effort for all other project activities. Secondary questions concern the possible effect of the above fact on the project cost and the quality of the inspections.

### 3.1. Measures

The measures that have been used for the post-mortem analysis were derived from two sources. The first was an activity reporting system in which all individuals were reporting their daily tasks by entering the time spent to each 'component'-'activity' combination and by indicating if this was normal working time or overtime. The integrity of the entries was secured by restrictions set by the database itself (checks for omitted or contradictory entries or not allowed combinations) and by cross checking of the reported data with the project progress and the personnel department data.

The second source of data for this study was the inspection record files of the project. Immediately after the completion of each inspection meeting, a formally structured inspection record (IR) was filled-in with information providing a more detailed view on the specific inspection. Information stored in the IR was the identity of the inspected item, the names and the roles of the participants, the inspection date, the document submission date, the inspection preparation time (sum of the inspectors' preparation time), the meeting time, and the descriptions of the discovered defects. To facilitate the statistical analysis of the data for the needs of this study, the fields of all project IRs were transferred to a corresponding database.

The data that has been collected for analyzing the project with respect to our research questions are the following:

a. *Project Effort per week*. Number of man-hours spent in all design activities—including inspection activities—during the course of the project. This measure is extracted from the activity reporting database and it is expected to provide an insight to the volume of work at several time points.

b. *Inspection Effort per week*. Number of man-hours spent in inspection related activities within the project. This measure is taken from the activity reporting database in order to determine periods during the project where inspection work has accumulated.

c. *Overtime Effort per week*. Number of man-hours reported as overtime work in the activity reporting system. The need for excessive inspection related work, in parallel to the usual design activities in a certain week could result in having some of the activities—either design related or inspection related—performed as overtime. Therefore, although the overtime work is not being spent necessarily to inspections, this measure will help to investigate any possible relationship between high inspection workload and overtime.

d. *Inspection Meeting Effort*. Total number of man-hours that all inspectors have spent for a certain inspection meeting. The measure is obtained from the IR database by multiplying the number of inspectors by the inspection meeting time.

e. *Inspection Preparation Effort*. Total number of man-hours that all inspectors spent in order to get prepared for a certain inspection. The *Inspection Preparation Effort* is the sum of the man-hours each inspector reported as preparation for the inspection meeting and it is explicitly written in the IR.

f. *Inspection Preparation Effort/Inspection Meeting Effort*. Since both the *Inspection Preparation Effort* and the *Inspection Meeting Effort* are directly related to the size of the inspected artifact, none of them can be compared to the corresponding value of a different inspection due to the differences in the size of the inspected items. The *ratio Inspection Preparation Effort/Inspection Meeting Effort* has been selected since it allows for comparisons between inspections of artifacts with varying sizes. To compare groups of inspections between different weeks, the individual ratios in 1 week are averaged.

g. *Required Design Period per component*. The total number of man-hours required for the completion of all the design activities related to a specific component as reported in the activity reporting system, divided by the number of normal working man-hours per week. In other words, this is the number of weeks that would have been spent for the completion of the work, if one designer was fully allocated to it.

h. *Actual Design Period per component*. Number of weeks that have elapsed between the start and finish date for each component as reported in the activity reporting system. Includes active period as well as periods during which the component was open (not finished) but idle (no man-hours declared for it). Large gaps (more than 2 weeks) during the development of the component are excluded.

i. *Stretch Ratio per component*. The ratio *Actual Design Period/Required Design Period*. This ratio is the factor by which the *Required Design Period* is stretched per component and indicates how much the development of each component is prolonged.

j. *Active components per week*. Number of components for which developers report working hours in a certain week. The measure is extracted from the activity reporting database.

### 3.2. Threats

#### 3.2.1. Threats to internal validity

As threats to internal validity we consider those factors that may cause interferences regarding the relationships that we are trying to investigate. Of the most important types of threats that apply to this kind of non-experimental studies [3,24] the following threats can pose risks to the applied research methodology.

*Incorrect model specification*. Important variables might have been ignored in the analysis. Since the collected data fits our primary and secondary hypotheses it can only be claimed that the discussed problems are consistent with the data from this case study.

*Reliability of procedures*. By this threat it is recognized that designers might have not applied the inspection related procedures in the correct manner. However, the fact that all designers have attended a formal training course on inspections, that each inspection team included an experienced software designer and that the whole process was closely monitored by the quality department, alleviates this threat.

*History effect*. The overall project work volume rather than the accumulation of inspections might have affected the dependent variables. To increase the validity of our results, the effort concerning all other project activities is also monitored, partially ensuring that project planning succeeded in achieving a smooth distribution of activities, thus reducing the pressure on inspection related activities.

*Statistical validity*. The performed analysis is based on some assumptions, which may affect the interpretation of the results. (a) To take into account the size of the inspected items, preparation time is divided by the duration of the review meeting. As a result, it is assumed that there is a positive correlation between size and meeting time. (b) It is expected that larger preparation and/or inspection times lead to increased number of discovered faults. These threats are valid; however, past experience increases the confidence that such assumptions are relatively safe to make.

*3.2.2. Threats to external validity*

As threats to external validity we consider those factors that limit the possibility to generalize our findings beyond the immediate case study to other settings and other times [24]. The performed analysis belongs to post-mortem pre-experimental designs and as such suffers from the criticism that such single cases offer poor basis for generalizing. The following threats belonging to the general category of 'Interaction of Setting and Treatment' threats [3], apply: difference of inspection process between companies; difference between project types; difference between programmer's capability.

Concerning the first threat, it has to be noted that the inspection methodology follows common industry practices and is based on Fagan's and Gilb's inspection method, as already described. Thus, since the inspection process resembles that of other organizations, similar results should be expected also in other environments.

The second threat is valid since this case study deals with a project from a specific domain. However, since telecommunications software development ranges high on both the technical and management complexity scale [17], the type of side-effects that are revealed should be useful to other kind of projects. Nevertheless, the intensity of the problems is expected to be higher in this specific project type.

Concerning the capability of the development team, it should be noted that this project employed personnel with varying experience levels. However, all project teams were guided by a team manager who coordinated the work of novice and experienced programmers, which is typical in the software industry.

## 4. Observations

Concerning the primary hypothesis of our analysis, the accumulation of software inspection effort in short time periods can be readily observed by a plot like the one shown in Fig. 1, which shows the *Project Effort per week* (Section 3.1.a) and *Inspection Effort per week* (Section 3.1.b) during the project life. It becomes obvious, that inspection effort presents spikes during the course of the project, although the normal project effort has a relatively smooth form, as a result of the proper scheduling of all other project activities. (The only exception to this smooth form of the project effort is during summer vacations, which reduce the workload abruptly). Moreover, the spikes in the inspection effort often are present in very short times and close to each other, possibly due to internal project deadlines.

To support the hypothesis that inspection effort is accumulated irrespectively of all other project activities, data concerning inspection and project effort are displayed in a different manner in Fig. 2. To produce the curves of Fig. 2, the values for *Inspection Effort per week* (solid line) and *Project Effort per week* (dotted line) have been sorted in descending order. The *y*-value of each point of the curves represents the percentage of the work that has been finished in the percentage of time denoted by the *x*-value. The purpose of this diagram is to illustrate that a certain percentage of the inspection effort is performed in a much shorter time period than the same percentage of the project effort, causing the spikes in Fig. 1 (67% of the inspection effort is finished in 20% of the time, while 67% of the project effort is finished in 45% of the time).

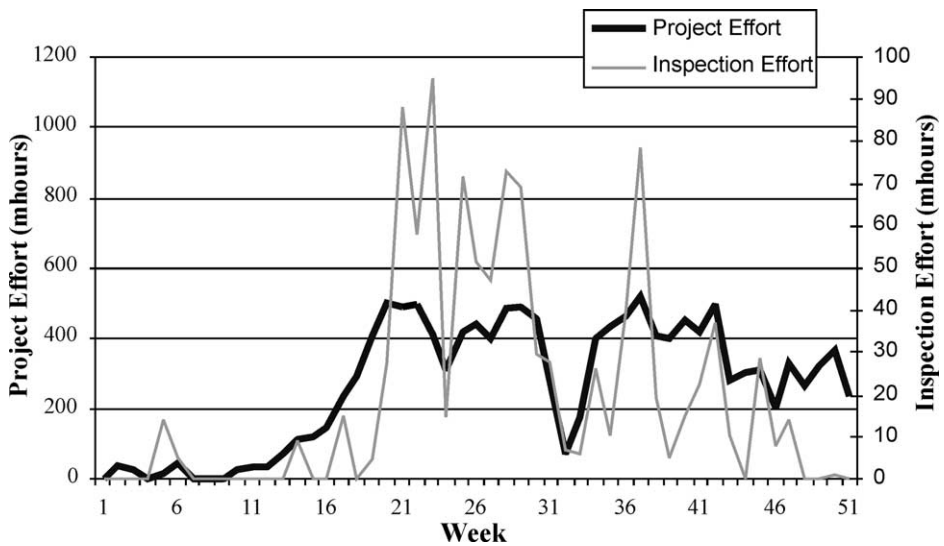


Fig. 1. Project effort (left axis) and inspection effort (right axis) versus project time. Inspection effort presents spikes during the course of the project, although the overall project effort has a relatively smooth form.

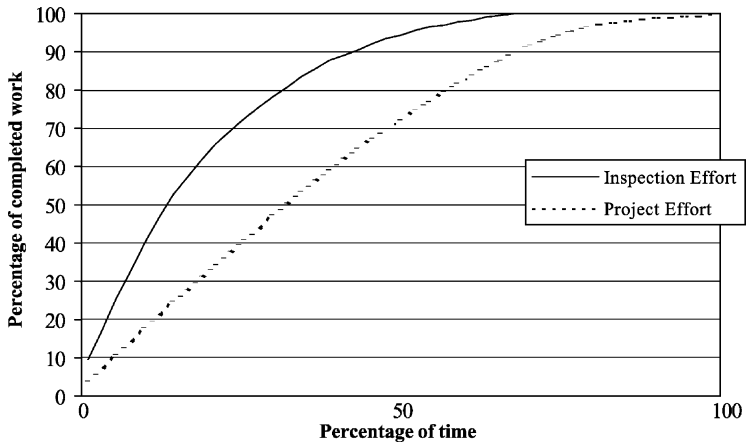


Fig. 2. Percentage of inspection and project effort versus percentage of time, when inspection and project effort per week have been sorted in descending order. A certain percentage of the inspection effort is performed in a much shorter time period than the same percentage of the project effort, causing the spikes in Fig. 1.

Our first hypothesis on the negative effect of accumulated inspections can be identified in Fig. 3, which reveals that as the *Inspection Effort per week* (Section 3.1.b) becomes larger, the *Overtime Effort per week* (Section 3.1.c) also increases (Pearson's correlation coefficient is equal to 0.689 and correlation is significant at the 0.01 level). A direct consequence of this positive correlation between *Inspection Effort per week* and *Overtime Effort per week*, is that in intervals with spikes in inspection related activities, overtime costs could become excessive, contributing significantly to the total project cost.

Even worse for the quality of the products that are being developed is the fact that when inspection effort accumulates, the relative preparation time for inspections decreases. This is illustrated in Fig. 4, which shows the correlation between the average ratio of *Inspection Preparation Effort/ Inspection Meeting Effort* per week (Section 3.1.f) over the sum of *Inspection Meeting Effort* per week (Section 3.1.d). (Correlation coefficient is equal to $-0.397$ and is significant at the 0.05 level). Since the sum of *Inspection Meeting Effort* per week is an indicator of the inspection work that has been accumulated, this plot validates the assumption that inspectors devote less time to preparation when effort presents spikes. Unfortunately, reduced preparation time has a profound impact on quality: recent studies have indicated that most defects are actually found during preparation for the inspections [13]. This is also in agreement with several studies [2,9], which report data confirming that reduced preparation and inspection time, inevitably cause quality degradation. Practically, what happens is that when preparation time is not sufficient, the preparation rate (inspected lines per engineer per hour) increases abruptly, leading to fewer major defects detected per KLOC [21].

In an effort to explore the possible reasons behind inspections aggregation, the development period for software components was analyzed. In Fig. 5, the horizontal bars correspond to the *Required Design Period* per component (Section 3.1.g), which is the required effort for each component, measured in man-weeks. In the same plot, the line indicates the *Stretch Ratio* per component (Section 3.1.i), which is a number presenting how longer the *Actual Design Period* (Section 3.1.h) per component compared to the *Required Design Period* is. The correlation coefficient

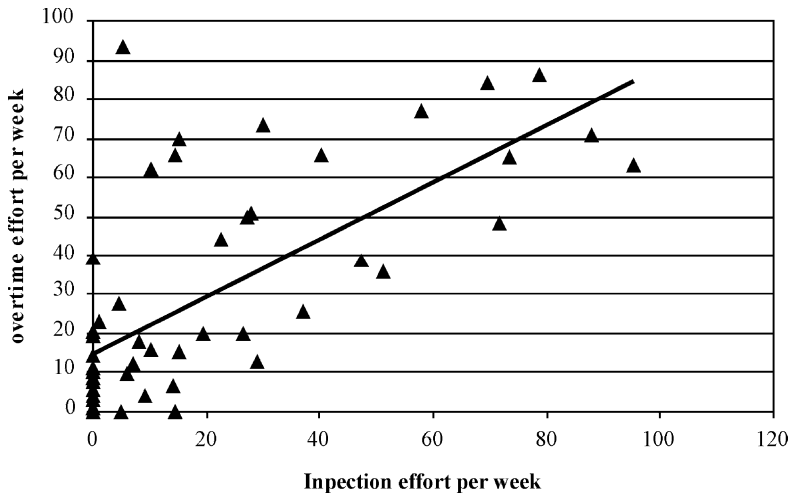


Fig. 3. Correlation between overtime effort and inspection effort per week.

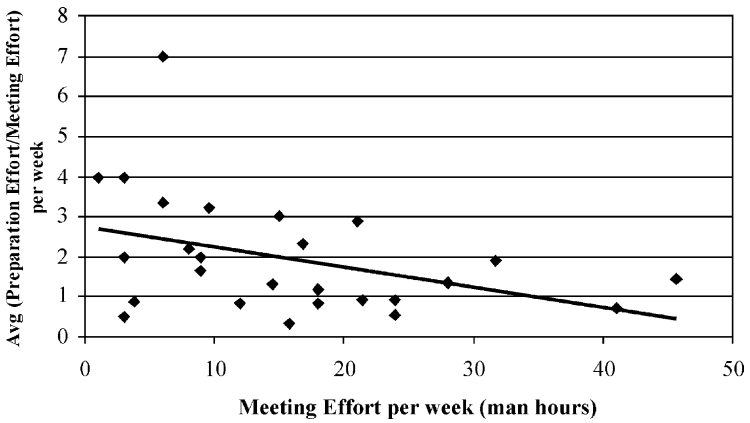


Fig. 4. Correlation between meeting effort per week and average preparation effort/meeting effort per week. When inspection meetings demand higher effort per week, the relative preparation time for inspections decreases.

between the values of *Required Design Period* and the values of *stretch ratio* is equal to −0.759 and is significant at the 0.01 level. Consequently, the development period is prolonged significantly more for components requiring less effort. This 'stretching' indicates that even if it were possible to review a large number of artifacts at points distributed over time, the development of all components has been prolonged and their finish date approaches that of large components (which unavoidably finish late) and consequently their inspection dates get closer.

For smaller components not only the design period is 'stretched' and prolonged but also the end of the design is moved as late as the deadlines allow. As it can be seen from Fig. 6 the *Active Components* per week (Section 3.1.j) gradually increase, as the project gets closer to the final deadline.

## 5. Discussion

Since inspection planning is not a central management process carried out during the initial stages of a project, a large number of factors affect the time in which inspection meetings are performed. Since it is extremely difficult to relate all possible factors to the inspection process by experimental results, the following are, according to the authors' view, also possible reasons that reduce inspection planning efficiency.

### *5.1. Weak coordination between project plan—inspection plan*

- Since a software project plan provides baseline cost and scheduling information that is required in order to begin the software engineering process [16], project planning is usually an activity that is performed early in a project's life cycle. On the contrary, inspection planning is performed later in the course of the project and in most cases when the first artifacts have already been developed. This distance in time between project and inspection planning does not allow for coordination and especially for taking inspection related resources and activities into account. As a result, the outcome of the overall project planning is a uniform distribution of effort for the most of the project's lifetime, while the lack of inspection planning activities causes inspections to accumulate in specific periods presenting spikes as already shown in Fig. 1.
- Inspection plans are set by the quality coordinator, while the project plan is written by the project manager. This

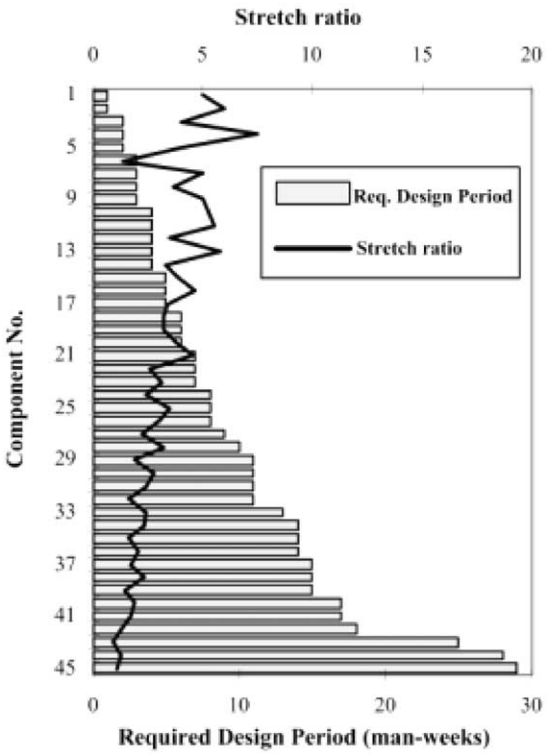


Fig. 5. The *required design period* per component (bars) (which is the number of weeks that would have been spent for the completion of each component's work if one designer was fully allocated to it) and the stretch ratio per component (line) (which shows how longer the actual design period per component is, compared to the required design period). The development period is prolonged significantly for components requiring less effort.

separation of roles presents a significant hindrance to the proper planning of inspections in relation to other project activities.
- Inspections are not taken into account during the construction of the overall project plan due to the intractability of the problem: the number of inspections, which results by multiplying the number of components, number of documents per component and number of project incremental phases, can be prohibitively large for a detailed and thorough plan.
- The project plan allocates resources on a component basis while the inspection plan is performed per artifact. Moreover, the development of a component might involve a number of programmers while each software artifact has a unique responsible person. These facts complicate even more the allocation of resources for inspection purposes and the alignment of inspection meetings with other activities.

### 5.2. Lack of estimates about the required capacity per week

- Even in cases when inspections are taken into account during initial planning activities, they are considered as milestones (a milestone being the end-point of a software process activity does not have any duration [18]). As a result, the actual effort for the inspections, which includes preparation time and inspection meetings, cannot be properly estimated and planned.
- Human resources, which take part in the inspections, are not centrally planned. Therefore, when inspections have to be performed, people participating in the inspection process are probably allocated to other time consuming tasks, limiting their involvement in both terms of time and effort to the preparation for reviewing a document.
- Similar to the previous observation is the fact that often, resources that participate in the inspection process, do not belong to the same project. For example, during the inspection of code, people from maintenance, system groups or quality assurance take part. Allocation and planning of people belonging to different projects or departments imposes a serious problem to efficient inspection planning.
- Inspections are usually not performed by setting a strict deadline by which the inspection meeting should be performed, but rather by a 'call to attend'. After this call by the moderator, the participants state freely the dates that are most convenient to them. This relaxed meeting scheduling limits the possibility to establish a central inspection plan, which will avoid multiple inspections during the same time. In addition, finding a suitable meeting time becomes harder as the number of attendees grows [15].

### 5.3. Project planning is adapted to the critical path

When project planning is performed, the task of the project manager is complicated by the fact that resources are usually not exclusively allocated to the project, but also participate in follow-up stages of previous projects or phases or are experts borrowed from other departments (e.g. maintenance). To keep up with project deadlines, resources are allocated to ensure that the critical path of the project has an acceptable finish date [23]. This is most often achieved by performing the initial scheduling starting from large components to meet the constraints imposed by the critical path and by assigning small components to partly allocated

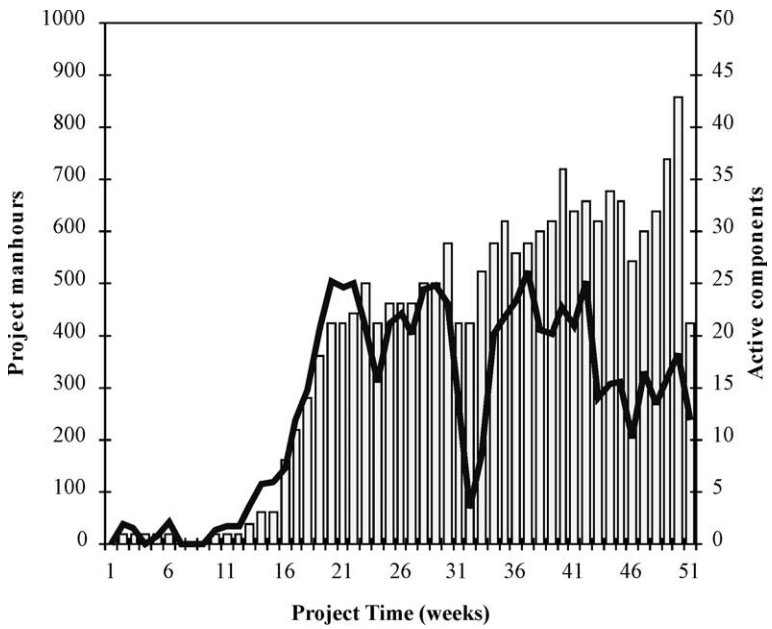


Fig. 6. Total project effort (continuous line—left axis) and number of 'active' components—components for which design hours have been reported in a certain week (bars—right axis). Active components gradually increase as the project approaches the final deadline.

designers. However, this is one of the most important causes for inefficient inspection planning. Although the artifacts corresponding to small components could have been finished well before those of other, larger components, this part-time allocation causes the development period for small components to be comparable to that of large components (Fig. 5).

On an individual basis, designers plan their tasks according to the approaching deadlines and not based on the inspections that have to be performed. Therefore, design documents are complete and ready for inspection close to project deadlines, where most other components are also finished (Fig. 6).

However, the possibility to review some artifacts before others is a theoretical one with many practical difficulties: development of all components should often be completed during the end of each project phase to include issues that have arisen from the development of other components. For example, improvements can be made by taking into account as much feedback from the maintenance department as possible. Moreover, technical issues can often be resolved by cross-checking similar products and this inter-product review can only be performed if their inspections are performed in parallel.

## 6. Proposed guidelines

Although it is relatively easy to identify the problems in a software development process caused from inefficient inspection planning, it is inversely difficult to suggest a step-by-step procedure to avoid them, due to the numerous parameters that are involved. However, a number of general guidelines can be derived from the observations made in Section 5, which are both easy to apply and have limited interference with other software development activities. These are:

- *The inspection plan must be ready as early as possible*. The existence of an inspection plan as early as possible, can contribute to securing control over the whole inspections procedure and to facilitating the execution of inspection related activities, since inspections are treated as intermediate component deadlines.
- *The required capacity for inspections must be secured*. Resources for inspection meetings and especially external experts should be secured from the beginning of the project to avoid lack of reviewers during inspections. Since external resources are usually booked on the basis of a standard allocation percentage per week, inspections should be planned in a way that makes use of this capacity.
- *The inspection plan must be coordinated with the project plan*. In order to avoid lack of inspection time or resources, the project plan should include an estimate of the inspection related effort per component (based on historical data and objectives) and establish a resource aggregation chart for the inspection activities. Inspections of critical artifacts should be viewed as tasks having duration and requiring allocated resources during the initial planning activities.
- *Design of small components should be finished as early as possible*. Small components, which are weakly coupled with the development of other products, should be finished as early as possible, in order to avoid congestion of inspections close to deadlines.
- *Inspections should be spread uniformly*. By getting assistance from all previous guidelines, an attempt to spread the inspections uniformly should be made. By achieving this, not only the accumulation of inspections in short periods is avoided but also alternative planning possibilities in case of unexpected situations exist.
- Finally, one approach to alleviate the problem of scheduling inspections during the course of a software project, could be based on the argument that meeting-less inspections are more efficient in detecting errors and more cost-effective than traditional meeting-based software inspections [14,22], especially when the inspection process is supported by appropriate infra-structure [12,20].

## 7. Conclusions

Data analysis for a large scale telecommunications software project has shown that for software inspections, which form an integral part in the software engineering process, the manner in which they are usually planned and performed, can give rise to a number of risks threatening the economics, quality, and the ability to meet deadlines of the project. Project planning methodologies, as currently applied in software project management, do not account for the inherent difficulties in planning software inspections and their related activities. As a result, inspection meetings accumulate at specific periods towards the project deadlines, possibly causing spikes in the project effort, overtime costs, quality degradation and difficulties in meeting milestones. The results provide a basis for discussing the reasons that drive inefficient inspection management and a number of guidelines that can be followed in order to coordinate inspections with all other project activities.

## References

[1] A.F. Ackerman, L.S. Buchwald, F.H. Lewski, Software inspections: an effective verification process, IEEE Software 6 (3) (1989) 31–36.
[2] J. Barnard, A. Price, Managing code inspection information, IEEE Software 11 (2) (1994) 59–69.
[3] T.D. Cook, D.T. Campbell, Quasi-Experimentation: Design and Analysis Issues in Field Settings, Rand McNally, Chicago, 1979.

[4] M. Fagan, Design and code inspections to reduce errors in program development, IBM Systems Journal 3 (1976) 219–248.
[5] T. Gilb, Principles of Software Engineering Management, Addison-Wesley, Reading, MA, 1988.
[6] T. Gilb, D. Graham, Software Inspection, Addison-Wesley, Reading, MA, 1993.
[7] P.M. Johnson, D. Tjahjono, Does every inspection really need a meeting?, Empirical Software Engineering 3 (1) (1998) 9–35.
[8] P.M. Johnson, Reengineering inspection, Communications of the ACM 41 (2) (1998) 49–52.
[9] R.T. McCann, How much code inspection is enough?, Crosstalk, The Journal of Defense Software Engineering 14 (7) (2001) 9–12.
[10] D. O'Neil, Issues in software inspection, IEEE Software 14 (1) (1997) 18–19.
[11] D. O'Neil, Peer Reviews, Encyclopedia of Software Engineering, Wiley, New York, 2001.
[12] D.E. Perry, A. Porter, M.W. Wade, L.G. Votta, J. Perpich, Reducing inspection interval in large-scale software development, IEEE Transactions on Software Engineering 28 (7) (2002) 695–705.
[13] A.A. Porter, L.G. Votta, V. Basili, Comparing detection methods for software requirements inspections: a replicated experiment, IEEE Transactions on Software Engineering 21 (6) (1995) 563–575.
[14] A. Porter, C.A. Toman, H. Siy,L.G. Votta, An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development, Proceedings Third ACM SIGSOFT Symposium on the Foundations of Software Engineering, Washington DC, 1995
[15] A.A. Porter, H.P. Siy, L.G. Votta, A review of software inspections, Advances in Computers 42 (1996) 40–76.
[16] R.S. Pressman, Software Engineering, McGraw-Hill, New York, 1997.
[17] W. Royce, Software project management: a unified framework, Addison-Wesley, Reading. MA, 1998.
[18] I. Sommerville, Software Engineering, Addison-Wesley, Harlow, UK, 1996.
[19] V. Sylaidis, D. Stasinos, T. Karvounidis, Better Telecommunications Software with Gilb's Inspection Method, Proceedings of Second International Conference on Product Focused Software Process improvement (PROFES'2000), Oulu, Finland, 2000.
[20] C.K. Tyran, J.F. George, Improving software inspections with group process support, Communications of the ACM 45 (9) (2002) 87–92.
[21] M. van Genuchten, C. van Dijk, H. Scholten, D. Vogel, Using group support systems for software inspections, IEEE Software 18 (3) (2001) 60–65.
[22] L.G. Votta, Does Every Inspection Need a Meeting?, Proceedings of First ACM Symposium on the Foundations of Software Engineering, Los Angeles, CA, 1993.
[23] R. Wild, Production and Operations Management, 5th ed., Cassell Educational Ltd, London, UK, 1995.
[24] R.K. Yin, Case Study Research: Design and Methods, SAGE Publications, Newbury Park, CA, 1989.