Journal of Universal Computer Science, vol. 18, no. 19 (2012), 2731-2746
submitted: 30/3/12, accepted: 6/9/12, appeared: 1/11/12 © J.UCS

A Global Software Inspection Process for Distributed Software Development

Deepti Mishra, Alok Mishra
(Department of Computer Engineering, Atilim University
Incek, 06836, Ankara, Turkey
deepti@atilim.edu.tr, alok@atilim.edu.tr)

Abstract: Globally distributed software development is an established trend towards delivering high-quality software to global users at lower costs. The main expected benefits from distributed software development are improvements in development time efficiency, being close to the customers and having flexible access to greater and less costly resources. Organizations require to use their existing resources as effectively as possible, and also need to employ resources on a global scale from different sites within the organization and from partner organization throughout the world. However, distributed software development particularly face communication and coordination problems due to spatial, temporal and cultural separation between team members. Ensuring quality issues in such projects is a significant issue. This paper presents global software inspection process in the distributed software development environment towards quality assurance and management.

Keywords: Global Software Development, Distributed Software Development, Software Inspection, Tool, Software quality
Categories: D.2.2, D.2.4

1 Introduction

Due to Globalization and Information and Communication Technologies (ICT) proliferation, Global Software Development (GSD) is increasingly pervasive. Many Organizations have turned to it in the quest for higher quality software delivered on time economically [Mishra and Mishra, 11]. Short term benefits of global software development are so appealing, more and more software companies begin globalizing their software development team and effort [Yu and Mishra, 10]. Software process desired result is high quality software at low cost [Mishra and Mishra, 09a]. Software inspection is a fundamental component of the software quality assurance process. The purpose of quality control task such as inspections, walkthroughs and reviews is for early and effective defect detection in order to improve product quality and reduce development rework [Ciolkowski et al., 03]. Software inspection is a structured, collaborative and established method of ensuring quality in software engineering. Capability Maturity Model Integration [CMMI, 02] level 3 organizations rountinely conduct work product assessments (i.e., inspections, walkthroughs and reviews) [Hale et al., 11]. In selecting a review approach, teams generally consider the trade-offs between the number of defects detected and the time and effort investment required (preparation and meeting time, scheduling delays, all multiplied by the number of reviewers) [Sauer et al., 00] [Votta, 93].

Distributed software development is a complex venture and distributed tasks have been proven to take up to 2.5 times more effort to complete than if the tasks were to have been done by co-located personnel [Herbsleb and Mockus, 03]. Traditional inspection processes cannot be simply adapted to be included into offshore or distributed software development where, large permanent companies are replaced by temporary group of developers collaborating on projects over the internet and in this context the inspection process must be supported by web based environment [Caivano et al., 01]. According to Laitenberger and Dreyer [98] there should be no substantial differences in efficiency between traditional and computer-supported inspections. Virtual software inspection is a process that conforms to a defined workflow and is performed in a distributed manner with the aid of an inspection tool. There are three significant aspects to be taken care of in virtual software inspection [Hedberg and Harjumma, 02]:

Tools that enable efficient running of the process. Independence of time and place, on-line recording of issues and data management can be achieved through network tools.

Flexibility of the process and supporting tools to ensure tolerable adoption effort and acceptance of the method.

Interoperability of the processes and tools, to enable convenient everyday use of the method and improves the effectiveness of inspections.

Tool support for software inspection evolved in 1990s and during this evolution the principles of distributed and asynchronous inspections were outlined [Hedberg, 04]. Instead of a fixed process model, virtual inspection tools should provide capabilities for customizing the process for an individual organization or project [Hedberg and Harjumma, 02]. The web based system facilitates support for distributed inspection in a virtual environment among global software development teams. Web technology facilitates the collaborative aspects of inspection as this not only introduces flexibility into the inspection meetings, but also enables easy, manageable distribution of the artefacts for inspection, including the document to be inspected, checklists, or any other related documents [Tervonen et al., 99]. In global software development, geographical distance becomes an augmenting factor for the costs of face-to-face meetings and the time distance can create barriers to the enactment of distributed virtual meetings conducted using ICT (e.g., text based chat, virtual blackboards, web platforms or virtual environments like Second Life) [Lucia et al., 11]. In order to overcome the issues related to performance of inspection processes in distributed settings, asynchronous discussions could be adopted before a face-to-face or virtual synchronous meeting [Damian et al., 08].

Virtual software inspection process can include asynchronous and synchronous phases through a network but conventional face to face meetings can be included if required. The synchronous activities of inspection include discussion of correlated faults, reaching a consensus on the faults, recording the action items, and determining the inspection's status [Mashayekhi et al., 93]. Teleconferencing and video-conferencing tools can be used for discussion purpose among participants. Traditional face-to-face discussions suffer from a number of process losses such as air-time fragmentation, blocking, evaluation apprehension, domination and free-riding [Nunamaker et al., 91]. Asynchronous computer-mediated communication systems tend to promote richer discussions than face-to-face exchanges but present additional

coordination challenges to team members working in this environment [Benbunan-Fich, 02]. Although asynchronous communication could be more efficient in promoting more carefully worded comments or more balanced participation, it could be less desirable due to the difficulty of conceptually integrating divergent contributions in order to produce the expected outcome [Nunamaker et al., 91]. The two main asynchronous activities of software inspection are the individual reviews and the producer's (authors of documents and codes) correlation of faults [Mashayekhi et al., 93].

An inspection tool is a software package particularly designed for inspection collaboration, and it should be capable of at least managing and delivering the inspection documentation on-line, enabling the effortless recording of defects and automatic gathering of defects [Hedberg and Harjumma, 02]. The objective of inspection is to locate potential defects (faults), not correct them. On-line inspection related material reduces paperwork, makes the latest material available to participants and thus facilitates in meetings. Material used in inspection includes the target material, the inspection-criteria list (check list), individual fault lists, the merged fault list, the action-item list, and the status report [Mashayekhi et al., 93]. The inspection information can be used for review and metrics collection to monitor the quality assurance. An inspection tool should support metrics and automate the collection, storage and analysis of the necessary data [Hedberg and Lappalainen, 05]. Hedberg and Lappalainen [05] further argued that to encourage process improvement, it must be possible to calculate the derived metrics automatically, and the set of metrics must be flexible enough to focus on the most important aspects of a given situation.

Globally distributed software development comes with several challenges related to different backgrounds of the partners, distance and time difference between sites [Pesola et al., 11]. Pesola et al. [11] further argued that the role of tools in distributed development is even higher than in single-site development because of the different background knowledge of people, e.g. about the product and its structure, and differences in the technical environment available in each participating site. To overcome these issues, a number of tools have been proposed for inspection planning [Aurum et al., 02], comment preparation [Bull, 97] and for both the individual preparation and the group meeting [Iniesta, 94]. Meyer [08] suggested to run the design and code review entirely on the web and desktop sharing solutions. A number of online inspection tools have been proposed in the past [Brothers et al., 90] [Gintell, 93] [Stein et al., 97].

The remainder of the paper is organized as follows: Section 2 presents related works of distributed software inspection tools. Section 3 describes global software inspection process. Section 4 provides details of global software inspection tool. Discussion is presented in section 5. Section 6 concludes with a summary and future work in this context.

2 Literature Review

In distributed software development, effective inspection process lead to increased correctness of analysis of results which is critical for success of the project. Based on Fagan's process Gintell et al. [93] introduced Scrutiny collaborative and distributed system for the inspection and review of textual software artefacts. Stein et al. [97]

found that distributed, asynchronous software inspection is feasible, cost effective means of collaboration for geographically distributed software development teams and suggested web-based tool Asynchronous Inspector of Software Artifacts (AISA) for such purpose. AISA was one of the first web-based software inspection tools which implemented Humphrey's model. This is also supported by Mashayekhi et al. [93] that cost-effectiveness of inspection would be improved further by a distributed collaborative meeting environment that eliminates the need for face to face meetings. They reported Collaborative Software Inspection (CSI) model to work from separate locations. According to Votta [93], Porter and Johnson [97], Miller et al. [98] and Sabaliauskaite et al. [04] face to face meetings do not improve the defect finding process, and suggested replacing meetings with other practices for instance, asynchronous discussion. Johnson and Tjahjono [98] introduced a controlled experiment in which they showed that the cost of a meeting is more than the cost of an asynchronous discussion. Knight and Meyers [91, 93] proposed an inspection technique that examines the artefacts in a series of small checklist-based inspection process.

Johnson [94] proposed Collaborative software review system (CSRS) flexible tool to support different software inspection processes by using a process modelling language for defining the process phases, the participant roles and the artefact to inspect. Asynchronous/Synchronous software inspection tool (ASSIST) by Johnson [94] like CSRS is designed to support any inspection process and any kind of software artefacts. This tool also provides an auto-collation facility to merge multiple list of issues or defects by using their similarity in terms of position, content and classification. E-mail notification is also included to support and inform the process. Perpich et al. [97] presented a web-based tool, named Hypercode, to asynchronously support distributed teams in the inspection of HTML documents. Tervonen et al. [98] introduced WiT (Web inspection Tool) towards virtual meetings and on-line recording of artefacts, checklists and other related documents.

Based on the analysis of 16 tools and their experience Hedberg and Harjumaa [02] concluded that flexibility and integration are two most significant features for implementing the next generation of inspection tools. According to Harjumaa et al. [01], there are two reasons for the full utilization of inspection software being extremely challenging: the variety of the inspection material qualities, and interfaces with other development tools and procedures. In most distributed inspection tools which are based on web, web services and servers are usually very limited and kept isolated from production system for security reasons along with a great deal of manual work towards control of an inspection tool [Harjumaa et al., 01]. Computerized software tools are the essence of the distributed software inspection process. Hedberg and Harjumaa [02] discussed the concept and features of virtual software inspections for distributed software projects and observed that document management for interoperability and mechanism for workflow control should be an integral part of the distributed software inspection tool. Hedberg and Harjumaa [02] proposed this virtual software inspections by implementing a new XML capable annotation tool, XATI that uses Mozilla as an application background to view the artefact under inspection. Jupiter, an inspection support tool developed as an eclipse plug-in was introduced by Yamashita [06]. The Jupiter tool only supports asynchronous discussion among inspectors, addresses the inspection of source code

but does not support distributed reviews. Caivano et al. [01] proposed Internet-Based Inspection System (IBIS) to support scalable and distributed software inspections which was further improved by Lanubile et al. [10] based on as variant of Fagan`s inspection process. Recently, Calefato and Lanubile [09] reported about EConference - a distributed conferencing system which can be used as collaboration tool for distributed meetings. Lucia et al. [11] proposed Advanced artefact management system (ADAMS) which integrates an artefact-based process support system for the management of human resources, projects and software artefacts with web-based artefact inspection tool (WAIT) for distributed inspection process.

Although this area has been studied intensively and numerous implementations exist, no tool has achieved a break-through. As distributed aspect has become more and more relevant in software development, therefore, the need for tools is now greater than ever [Hedberg, 04]. Here, we have extended our previous work [Mishra and Mishra, 10] [Mishra and Mishra, 09b] [Mishra and Mishra, 07] by including global software inspection process and tool to provide effective means for geographically distributed software development groups.

3 Global Software Inspection Process

Currently an updated and improved version of global software inspection process is used by automating the inspection and meeting processes. Various stages of global software inspection process are shown in figure 1.

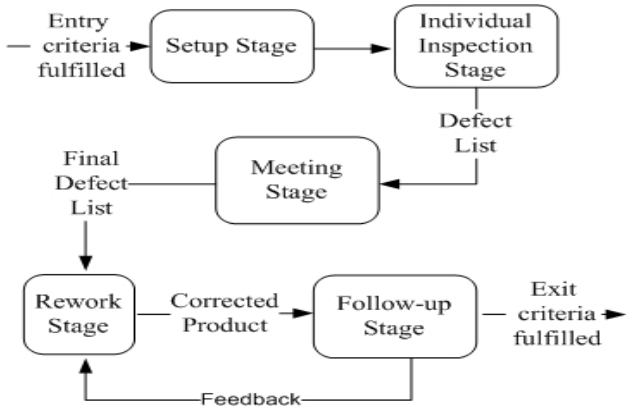

Figure 1: Stages in Global Software Inspection Process [Mishra and Mishra, 10]

The inspection process begins when entry criteria are satisfied. The main entry criterion is that the product to be inspected is complete and mature enough to be used after the defects will be removed. The author informs the software quality team leader about the completion of the product that will be inspected.

Setup Stage: In the setup stage, the inspection team leader selects the members of the inspection team and generates an inspection plan. Then, the document to be inspected as well as other necessary documents i.e. checklists, are uploaded on the

tool by the leader. The leader can also send an email to the members regarding the details of the planned inspection that also includes their responsibility, deadlines etc., via the tool. The leader can also put an announcement consisting of these details on the tool itself.

Individual Inspection Stage: Inspectors inspect the product independently with the help of checklists provided in the tool and store their comments on the web-based tool. Inspection is done according to the checklists appropriate for the inspected product, like the code review checklist, requirements inspection checklist or design review checklist. These checklists are available to inspectors in the tool. Inspectors cannot see each others comments because it may influence them. The inspection team leader can see all comments entered by every inspector.

Meeting Stage: In this stage, all inspectors, including the leader, get together to have online inspection meeting via the tool. The timing of the meeting is intimated to the team by the leader either by via e-mail or by posting an announcement. They discuss defects they have found during the individual inspection stage. These discussions help in identifying the true defects and eliminating the false defects from the defect list. Then a final defect list is made by the leader which is then emailed to the author.

Rework Stage: In this stage, the author of the product performs a rework over the materials to correct them. The author updates the product according to the final defect list and takes notes next to every defect explaining what changes have been done along with their locations.

Follow-up Stage: The inspection team leader or one of the inspectors performs a follow-up to ensure that every issue is addressed and every defect is corrected. If all defects are not removed, the product is given back to the author to correct them, so the product goes back to Rework Stage.

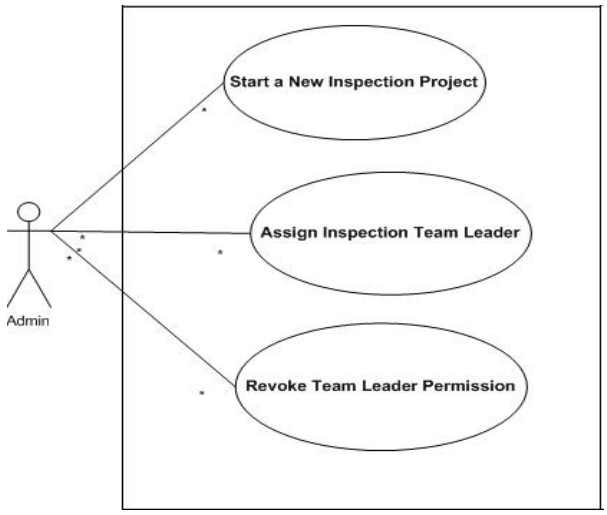

Figure 2: Use-cases for Admin

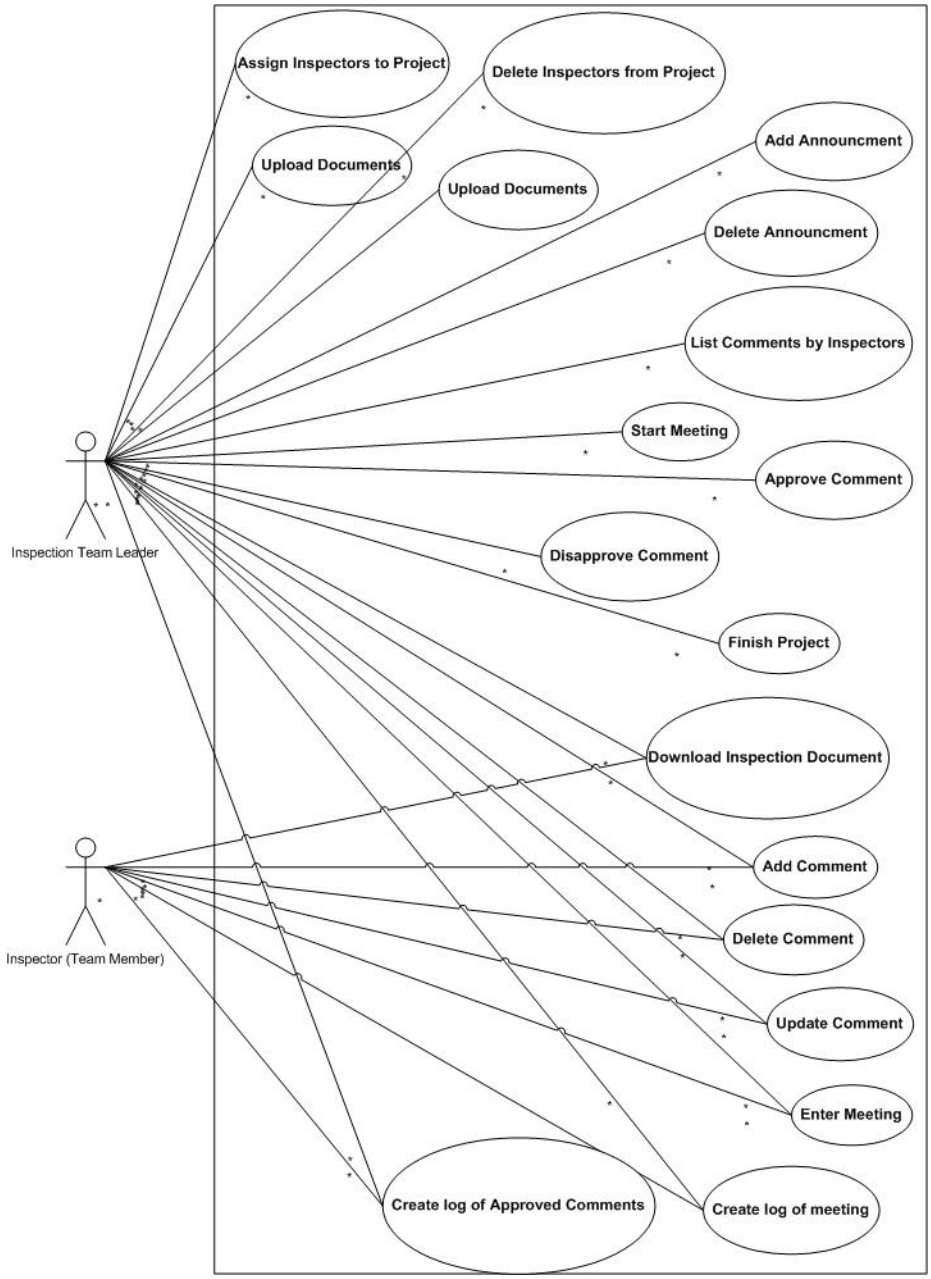

Figure 3: Use-cases for Inspection Team Leader and Members

4 Global Software Inspection Tool

The global software inspection process was automated by developing a tool as shown in figure 2. This tool is developed with PHP, MySQL, and Apache Server. The primary elements are termed as "actors", and the processes are termed as "use cases". There are three types of actors: admin, inspection team leader, and inspectors and their use cases are shown in figure 2 and 3.

Admin will log on to the web-based tool and start a new inspection project as shown in figure 4. Then the admin has to assign a team leader from the existing staff for the inspection team. Staff members can register to the system by themselves and their information along with the email is stored during the registration process. Now, the person chosen as the team leader has additional permission. These permissions can be taken back by the admin once the inspection is finished.

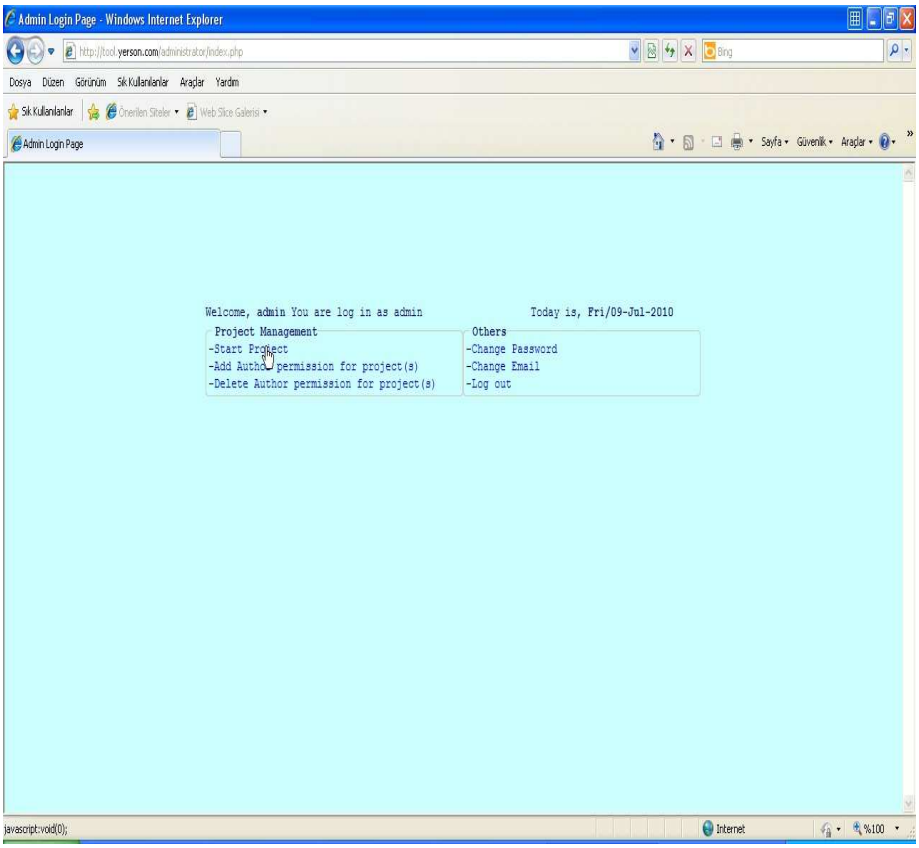

Figure 4: Admin screen

Team leader will now logon to the system and can see the project already on the welcome screen. Team leader can choose people to work as inspectors as shown in figure 5. All the necessary documents (e.g. checklists etc.) along with the document to

be inspected is uploaded by the team leader. Team can announce details about the inspection (schedule, responsibilities etc.) by sending a mail to all inspectors chosen. Also, this information can be displayed by adding an announcement which will be seen by all inspectors as soon as they will logon to the system. Announcement which are no longer valid may be deleted by the team leader later. Announcements deleted by the team leader will be automatically deleted from the inspectors screens.

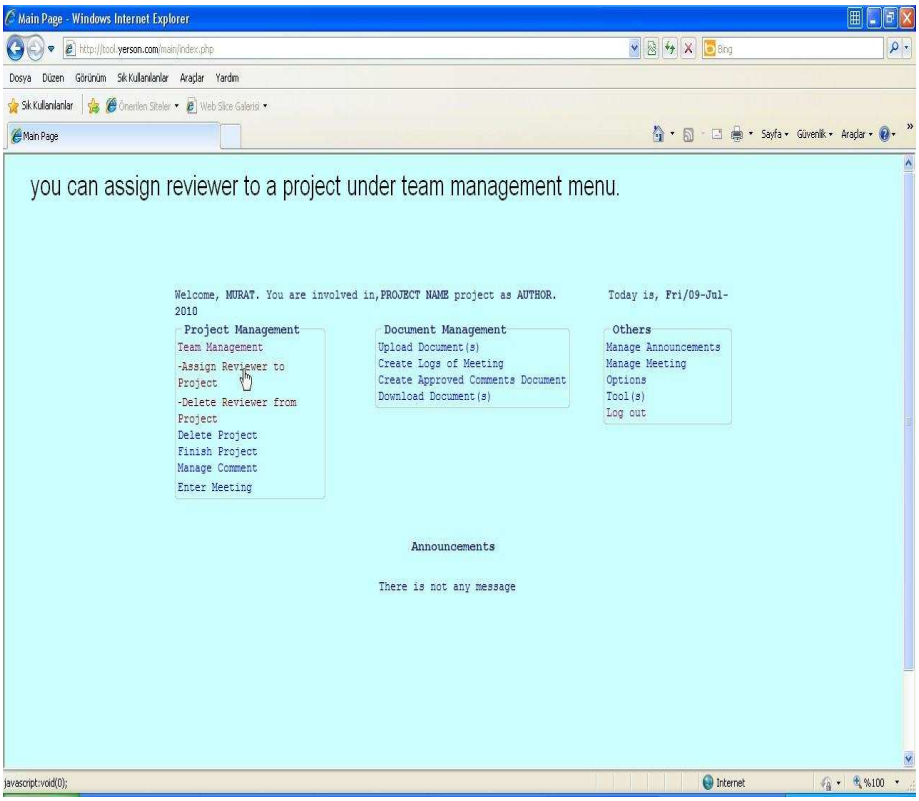

Figure 5: Team leader screen

Inspectors can log on to the system and can see all the documents related with the inspection. They can give add comments (potential problems in the document), update their comments and delete the unnecessary comments. All the comments entered by different inspectors can be seen by the team leader only. After all inspectors finish the individual inspection, the team leader can start the online inspection meeting as shown in figure 6. All the comments can be discussed here to find the actual problems. Team leader can approve or disapprove a comment based on the discussions. All the approved comments are true defects. The team leader as well as team members can store all discussions held between inspection team members by creating the log of the meeting. Similarly, a log of all the approved comments can also be created by the team leader. If an inspector can not attend the online meeting, he\she

can still get these details. Team leader can create the log of all approved comments and send it to the author. Author will do the rework to address all comments. The updated product is uploaded by the team leader and one of the inspection team member checks whether the revised product has addressed all the comments. If yes, then inspection process is finished otherwise it will be again sent to the author for rework. If the inspection process is finished, team leader will click on finish inspection project. The team leader is responsible for evaluating, reporting and follow-up activities, whereas the author does the editing. The edit and follow-up phases are important to assurance that defects will be corrected.

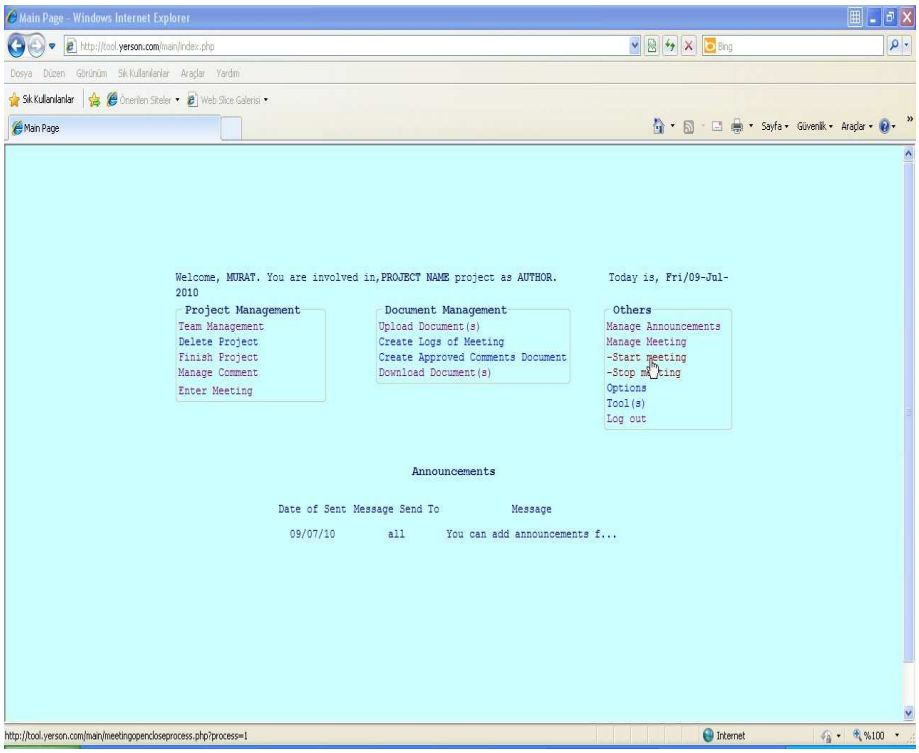

Figure 6: Team leader screen

MacDonald [98] summarized a list of features of software inspection tools, including linked annotation, defect classification, cross-referencing, automated analysis, checklists, supporting material, distributed meetings, decision support, and data collection. Sapsomboon [00] classified these features into three broad categories: inspection functions, support functions, and availability. Table 1 compares recent tools with the introduced web based software inspection tool in this study.

	AISA	ASSIST	CAIS	CSI	EMS	IBIS	ICICLE	SCRUTINY	INSPECTA	WIP	XATI	GIT
Inspection Functions												
Linked Annotation	●	●	●	●			●	●	●	●	●	
Defect Classification	●	●	●	●		●	●	●		●	●	●
Cross-referencing		●					●					
Data Analysis												
- Data Collection		●	●	●		●	●	●			●	
- Automated Analysis				●			●				●	
Support Functions												
Checklists		●	●	●		●			●	●		●
Reference Material		●			●	●	●		●	●		●
Meeting Support												
- Distributed Meeting		●		●	●	●		●			●	●
- Synchronous Facility		●		●		●	●	●			●	●
- Process Support			●		●	●						●
- Voting Facility	●	●	●		●			●	●			
- Discussion Thread			●			●			●			●
Scheduling Support												
- Scheduling					●							●
- Email Notification		●		●	●				●			●
Decision Support	●	●	●					●				
Availability												
Document Support						●					●	●
- Graphical Document	●										●	●
WWW-based	●					●				●	●	●
Cross-platform	●					●				●	●	●

Table 1: Comparison of different tools with GIT tool

5 Discussion

The inspection tools of web generation emphasize the benefits of hypertext and structured documents, but still not many organizations have adopted and used these features. Before these advantages can be capitalized on, organizational change must happen, so the evolution of the software inspection tools and process improvement they provide is closely linked to the process which produces artefacts for inspection

[Hedberg and Lappalainen, 05]. The comparison between these inspection tools is interesting and above table 1 provides this on various functions of the tools. The main objective of the software inspection is to locate potential defects in the artefact and this NEW tool (GIT-Global Inspection Tool) is provides defect classification. This tool also facilitates most of the support functions like checklists, reference material, Distributed Meeting, Synchronous Facility, Process Support, and Discussion Thread. In terms of scheduling support it includes scheduling and e-mail notification services. Document Support, Graphical Document, WWW-based and Cross-platform features are also part of this tool which enhances usability and availability of the tool on various platforms. The main limitation of the off-the-shelf document producing tools are in the area of metrics collection and process improvement support. Hedberg and Lappalainen [05] observed that present tools fully achieve the acceptance levels in the artefact management and quality areas, meaning that inspections can be carried out with the help of these tools. It is interesting to note by Lucia et al. [11] that despite the available number of distributed inspection processes and tools, the industrial practice is still far to adopt them since the management consider them non-effective. They also argued that lack of integrated environments to support all the phases of a development process. Distributed inspection tools are not widely popular and used in software industrial environment although these tools are being recognised useful to improve quality. A collaborative development environments (CDE) provides a project workspace with a standardized tool set for global software team. No current tool or CDE supports all the activities for global software development and users must therefore prioritize their collaboration requirements and tools to support them [Lanubile et al., 10]. CDEs combine several of the tools to provide smooth development environment to increase developer comfort and productivity [Booch and Brown, 03]. Lanubile et al., [10] further suggested that effective tool support for collaboration is a strategic initiative for any company with distributed resources and this is the only way to perform this efficiently, consistently, and securely. Hedberg and Lappalainen [05] suggested an evaluation criteria based on the functional requirements of a software inspection tool. These are divided into five categories: artefact management, defect and process management, as well as process improvement support and quality aspects. Infact this is an extension of earlier evaluations [Macdonald, 1995, Tenhunen and Sajaniemi, 02] along with aspects of virtual inspection [Harjumaa et al., 01].

As a prelimanary survey after deployment of tool in software organization among practitioners observations are summarized as following:

- This model is asynchronous. Inspectors inspect the product or part of the product independently without coming together at one place and send their comments via a web-based tool.
- Inspection meeting is done online through the tool without coming together physically. If an inspector can not login during the meeting time, he/she can still download the log of the meeting to know the details about the meeting.
- This inspection method is automated by developing a web-based tool so it eliminates lots of labour-intensive paperwork. Total inspection and meeting time is reduced, people resource is saved. Paper usage is reduced towards green computing.

- Due to the usage of tool in this inspection method, getting the inspection data from the past projects is easier. This data can be helpful for the estimation of time, cost and resource for inspection in future projects. Also it can be used to further improve the inspection process.
- All the checklists are available in the tool which helps inspectors in terms of efficiency and productivity.
- This inspection process includes early life cycle artifacts (for example, requirements) along with inspection of code.
- E-mail notifications to enhance context knowledge within an inspection process.

The constraints obeserved by the practitioners are:

- Many studies suggest that face-to-face meeting is best to find defects in complex software development problems. In the proposed process, although meeting is done online with the help of tool but, if required, face-to-face meeting can be organized.
- Data collection and automated analysis will assist towards metrics measurement.

6 Conclusion

Due to the proliferation of distributed software development, the role of virtual software inspection will be more significant in the future. Distributed software development projects can not make use of traditional methods although their communication and quality assurance needs are the same. Integration with data repositories, project and version management system will enhance the importance of software inspection. Web technology facilitates the collaborative aspects of inspection. Apart from the flexibility of the inspection meetings it also enables easy, manageable distribution of the artifacts for inspection, including checklists and other related documents. The proposed global software inspection process with tool support has been initiated towards deployment in a software organization. As a future work it is planned to compare this process and tool support with existing distributed inspection process towards further improvement by employing as case studies in different software organizations. Also, an empirical evaluation with software professionals to evaluate its effectiness in supporting the distributed inspection process is planned in the organization.

References

[Aurum et al., 02] Aurum, A., Petersson, H., Wohlin, C.: 'State-of-the-art: software inspections after 25 years', Softw. Test. Verif. Reliab., 2002, 12, (3), pp. 133–154

[Benbunan-Fich, 02] Benbunan-Fich R., Hiltz, S.R. and Turoff, M.: A comparative content analysis of face-to-face vs. asynchronous group decision making, Decision Support Systems, 34(2002), pp. 457-469.

[Booch and Brown, 03] G. Booch and A.W. Brown: Collaborative Development Environments, *Advances in Computers,* vol. 59, 2003, pp. 2–29.

[Brothers et al., 90] Brothers, L.R., Sembugamoorthy, V., Muller, M.: 'ICICLE: Groupware for code inspections'. Proc. 1990 ACM Conf. Computer Supported Cooperative Work, 1990, pp. 169–181.

[Bull, 97] Bull. Inspection Process Assistant: User Guide, 1997

[Caivano et al., 01] Caivano, D., Lanubile, F., Visaggio, G.: Scaling up Distributed Software Inspection Proceedings of the ICSE Workshop on Software Engineering over the Internet, available at http://sern.ucalgary.ca/~maurer/icse2001ws/submiss

[Calefato and Lanubile, 09] Calefato, F. and Lanubile, F.: Using frameworks to develop a distributed conferencing system: an experience report. *Softw. Pract. Exper.* 39, 15 (Oct. 2009), pp. 1293-1311.

[Ciolkowski et al., 03] Ciolkowski, M., O. Laitenberger, & S. Biffl, : Software reviews, the state of the practice, *IEEE Software,* (20:6), pp. 46-51.

[CMMI, 02] CMMI Product Team: Capability Maturity Model Integration (CMMISM) Version 1.1, Software Engineering Institute, CMU/SEI2002- TR-012. Pittsburg, PA.

[Damian et al., 08] Damian, D., Lanubile, F., Mallardo, T.: On the need for mixed media in distributed requirements negotiations, IEEE Trans. Softw. Engng., 2008, 34, (1), pp. 116–132

[Gintell, 93] Gintell, J.W., Arnold, J., Houde, M., Kruszelnicki, J., McKenney, R., Memmi, G.: 'Scrutiny: a collaborative inspection and review system'. Proc. European Conf. Software Engineering, 1993, pp. 344–360.

[Hale et al., 11] David P. Hale, Joanne E. Hale, and Randy K. Smith: Evaluation of work product defects during corrective & enhancive software evolution: a field study comparison. *SIGMIS Database* 42, 1 (February 2011), pp. 59-73.

[Harjumaa et al., 01] Harjumaa, L., Hedberg, H., and Tervonen, I.: A Path to Virtual Software Inspection. In *Proceedings of the Second Asia-Pacific Conference on Quality Software* (December 10 - 11, 2001). APAQS. IEEE Computer Society, Washington, DC, 283.

[Hedberg, 04] Hedberg, H.: Introducing the Next Generation of Software Inspection Tools, Proceedings of 5th International Conference of Product Focused Software Process Improvement (PROFES 2004), Springer Verlag, 234-247.

[Hedberg and Harjumma, 02] Hedberg, H. and Harjumma, L.: Virtual Software Inspections for Distributed Software Engineering Projects, Proceedings of ICSE International Workshop on Global Software Development, available at
http://www.tol.oulu.fi/i3/2002/hedberg_icse_2002.pdf

[Hedberg and Lappalainen, 05] Hedberg, H. and Lappalainen, J.: A Preliminary Evaluation of Software Inspection Tools, with the DESMET Method, Proceedings of the Fifth International Conference on Quality Software (QSIC'05), IEEE Computer Society, pp. 45-54.

[Herbsleb and Mockus, 03] Herbsleb, J. D. and Mockus, A.: An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Trans. Softw. Eng.* 29, 6 (Jun. 2003), 481-494. DOI= http://dx.doi.org/10.1109/TSE.2003.1205177

[Iniesta, 94] Iniesta, J.B.: 'A tool and a set of metrics to support technical reviews', in Ross, M., et al. (ed.): 'Software Quality Management II, Volume II: Building Quality into Software' (Computational Mechanics, Southampton, UK, 1994), pp. 579–594.

[Johnson, 94] Johnson, P.M.: 'An instrumented approach to improving software quality through formal technical review'. Proc. Int. Conf. Software Engineering, 1994, pp. 113–122.

[Johnson and Tjahjono, 98] Johnson, P. M. and Tjahjono, D.: Does Every Inspection Really Need a Meeting?. *Empirical Softw. Engg.* 3, 1 (Jul. 1998), 9-35. DOI= http://dx.doi.org/10.1023/A:1009787822215

[Knight and Meyers, 91] Knight, J.C., Meyers, E.A.: 'Phased inspections and their implementation', Softw. Engng. Notes, 1991, 16, (3), pp. 29–35.

[Knight and Meyers, 93] Knight, J.C., Meyers, E.A.: 'An improved inspection technique', Commun. ACM, 1993, 36, (11), pp. 51–61.

[Laitenberger and Dreyer, 98] Laitenberger, O. and Dreyer, H. M.: Evaluating the Usefulness and the Ease of Use of a Web-based Inspection Data Collection Tool. In *Proceedings of the 5th international Symposium on Software Metrics* (March 20 - 21, 1998). METRICS. IEEE Computer Society, Washington, DC, 122.

[Lanubile et al., 10] Filippo Lanubile, Christof Ebert, Rafael Prikladnicki, Aurora Vizcaíno: Collaboration Tools for Global Software Engineering, *IEEE Software*, vol. 27, no. 2, pp. 52-55, March-April 2010.

[Lucia et al. ,11] De Lucia, A., Fasano, F., Scanniello, G., Tortora, G.: Improving artefact quality management in advanced artefact management system with distributed inspection. IET Software 5(6): pp. 510-527.

[MacDonald, 98] MacDonald, F.: Computer Supported Software Inspection. Department of Computer Science, University of Strathclyde, UK (1998).

[Macdonald et al., 95] Macdonald, F., Miller, J., Brooks, A., Roper, M, and Wood, M.: A Review of Tool Support for Software Inspection. In *Proceedings of the Seventh International Workshop on Computer-Aided Software Engineering* (CASE '95). IEEE Computer Society, Washington, DC, USA, pp. 340-349.

[Mashayekhi et al., 93] Mashayekhi, V., Drake, J. M., Tsai, W., and Riedl, J.: Distributed, Collaborative Software Inspection. *IEEE Softw.* 10, 5 (Sep. 1993), pp. 66-75.

[Meyer, 08] Meyer, B.: 'Design and code reviews in the age of the internet', Commun. ACM, 2008, 51, (9), pp. 67–71.

[Miller et al., 98] Miller, J.,Wood, M., Roper, M.: 'Further experiences with scenarios and checklists', Empirical Softw. Engng., 1998, 3, (1), pp. 37–64.

[Mishra and Mishra, 11] Mishra, D. and Mishra, A.: A review of non-technical issues in global software development. International Journal of Computer Applications in Technology, 2011 Vol.40, No.3, pp. 216 - 224

[Mishra and Mishra, 10] Mishra, D. and Mishra, A.: A software inspection process for globally distributed teams. In *Proceedings of the 2010 international conference on On the move to meaningful internet systems* (OTM'10), Robert Meersman, Tharam Dillon, and Pilar Herrero (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 289-296.

[Mishra and Mishra, 09a] Mishra, D., Mishra, A.: Software Process Improvement in SMEs : A Comparative View. Computer Science and Information Systems, Vol. 6, No. 1, pp. 111-140.

[Mishra and Mishra, 09b] Mishra, D. and Mishra, A.: Simplified software inspection process in compliance with international standards. Computer Standards and Interfaces, 31, 4 (Jun. 2009), pp. 763-771.

[Mishra and Mishra, 07] Mishra, D., and Mishra, A.: An efficient software review process for small and medium enterprises**.** IET Software, 1(4), pp. 132-142

[Nunamaker et al., 91] Nunamaker, J., Dennis, A., Valacich, J., Vogel, D., & George, J.: Electronic meeting systems to support group work. Communications of the ACM, 34 (7), pp. 41-61.

[Perpich et al., 97] Perpich, J.M., Perry, D.E., Porter, A.A., Votta, L.G., Wade, M.W.: 'Anywhere, anytime code inspections: using the web to remove inspection bottlenecks in large-scale software development'. Proc. Int. Conf. Software Engineering, 1997, pp. 14–21.

[Pesola et al., 11] Pesola, Jukka-Pekka, Tanner, Hannu, Eskeli, Juho, Parviainen, Päivi, Bendas, Dan: Integrating Early V&V Support to a GSE Tool Integration Platform, icgse-w, pp.95-101, 2011 IEEE Sixth International Conference on Global Software Engineering Workshop, 2011

[Porter and Johnson, 97] Porter, A.A., Johnson, P.M.: 'Assessing software review meetings: results of a comparative analysis of two experimental studies', IEEE Trans. Softw. Engng., 1997, 23, (3), pp. 129–145.

[Sabaliauskaite et al., 04] Sabaliauskaite, G., Kusumoto, S., Inoue, K.K.: 'Assessing defect detection performance of interacting teams in object-oriented design inspection', Inf. Softw. Technol., 2004, 46, (13), pp. 875–886.

[Sapsomboon, 00] Sapsomboon, B.: Shared defect detection: The Effects of Annotations in Asynchronous Software Inspection, Faculty of Information Sciences, University of Pittsburgh (2000).

[Sauer et al., 00] Sauer, C., D.R. Jeffery, L. Land, & P. Yetton: "The effectiveness of software development technical reviews: a behaviorally motivated program of research," *IEEE Transactions on Software Engineering,* (26:1), pp. 1-14.

[Stein et al., 97] Stein, M., Riedl, J., Harner, S. J., and Mashayekhi, V.: A case study of distributed, asynchronous software inspection. In *Proceedings of the 19th international Conference on Software Engineering* (Boston, Massachusetts, United States, May 17 - 23, 1997). ICSE '97. ACM, New York, NY, 107-117

[Tenhunen and Sajaniemi, 02] Tenhunen V. and Sajaniemi, J. An Evaluation of Inspection Automation Tools. In *Proceedings of the 7th International Conference on Software Quality* (ECSQ '02), Jyrki Kontio and Reidar Conradi (Eds.). Springer-Verlag, London, UK, UK, pp. 351-362.

[Tervonen et al., 98] Tervonen, I., Iisakka, J., Harjumma, L. (1998) Software Inspection- a blend of discipline and flexibility, Proceedings of ESCOM-ENCRESS conference, pp. 157-166.

[Votta, 93] Votta, L.G.: "Does every inspection need a meeting?," *First ACM SIGSOFT Symposium on the Foundations of Software Engineering,* ACM, Los Angeles, pp. 107-114.

[Yamashita, 06] Yamashita, T.: 'Evaluation of Jupiter: a lightweight code review framework'. MS thesis, University of Hawaii, 2006, drafted from csdl.ics.hawaii.edu/techreports/06-09/06-09.pdf

[Yu and Mishra, 10] Yu, L. and Mishra, A.: Risk Analysis of Global Software Development and Proposed Solutions, AUTOMATIKA 51(2010) 1, pp. 89–98.