DRAFT

Full paper published in

International Transactions on Systems Science and Applications, Volume 1, No. 2 (2006)

Empirical Evaluation of Reading Techniques for UML Models Inspection

Aleksander Jarzębowicz[1], Janusz Górski[1]

[1] Gdańsk University of Technology, Dept. of Software Engineering
Narutowicza 11/12, 80-952 Gdańsk, Poland
Email: {olek, jango}@eti.pg.gda.pl

Abstract: This paper reports on an experiment comparing three reading techniques: ad hoc, UML-HAZOP and scenario-based applied during inspection of UML models. UML-HAZOP is a reading technique adopted from the domain of safety analysis of critical systems. The technique and its variants applied in experiment are outlined in the paper. Design, a way of conducting and results processing of the experiment are described. The results include comparison of effectiveness and efficiency (for pairs) and analysis of number of reported defects as a function of time. The experiment revealed that although pair effectiveness was rather similar for all techniques, for pair efficiency UML-HAZOP had the best performance.

Keywords: software inspection, empirical software engineering, controlled experiments, object-oriented design, unified modeling language.

1. Introduction

Software inspections are a well-known tool for improving quality and decreasing cost related to software development, and are in use for thirty years now (as Fagan's initial work was reported in 1976) [6]. The scope of inspections is not limited to software code and also includes other software artifacts which can contain defects that, if not detected and removed, can require significant rework effort. In particular, the artifacts created during early phases of software development are considered as an attractive target for inspections because their defects, if not removed early, can be extremely costly.

In our research we focus on UML (Unified Modeling Language) [23], [24] models because they are widely applied in industrial practice. Such models are developed in early phases and have a direct influence on the resulting software (especially when applying forward-engineering supported by CASE tools). Several software development methodologies (e.g. Rational Unified Process) consider them as an essential product of development process. UML models are situated in between the requirements and code which were of major interest to the inspection processes, while inspecting UML (or other) models is addressed in less degree. In our research we aim at development of a dedicated inspection method for UML models.

Since 1976 there were many suggestions of how to improve inspections to make them more effective and efficient. Some attempts concerned changes in an inspection process e.g. further formalization and use of measurement [7], addition of brainstorming meeting [7], reducing number of participants [2], removing group meeting from the process [30], dividing inspection into sequential phases [14]. Others suggested optimization of checklists to better tailor them to a given application context. For instance object-oriented languages are prone to several categories of defects that were not present for procedural languages [5].

Checklists have been used since the beginning of software inspections and still are widely used in industry. It is however argued that checklists do not focus inspector's attention in a sufficient way as they only require passive

reading and a better idea is a reading technique which stimulates a more active work of the inspector [25]. A large industrial study conducted to determine which factors influence inspection effectiveness, proved that the most influencing factor was the reading technique applied [26].

Several proposals of alternative reading techniques were already published. Usually, they apply checklists but in addition provide scenarios to be followed by inspector. Scenarios contain additional tasks aiming at better understanding of the inspected artifact. In Defect-Based Reading (DBR) technique [27], designed for inspecting requirements specifications, scenarios are based on defect taxonomies. Perspective-Based Reading (PBR) [1] was designed for inspecting requirements and code. This technique takes the perspective of the further user of the inspected artifact (e.g. designer, tester). Use-Based Reading (UBR) [32] focuses on inspecting user interfaces. New reading techniques have been thoroughly evaluated by series of controlled experiments conducted both in academic [3], [19] as well as in industrial [1], [17] environments.

Inspections of UML models also became an object of interest of some researchers. A scenario-based reading technique named Traceability-Based Reading (TBR) has been proposed [15], [18], [29] and empirically evaluated [16]. Later versions of this technique are also known under name Object-Oriented Reading Technique (OORT) [4]. This work is closely related to ours, however the focus is on different defect category. OORT is focusing mainly on inconsistencies between various diagrams being parts of the same model. Although we consider consistency of the model an important issue, we have not followed this direction. The basic consistency checks are usually done by CASE tools which are continuously becoming more sophisticated. Other consistency issues can be addressed by dedicated automated tools (although it needs more formal representation of UML semantics [11]). In our work we concentrated on other category of defects – erroneous contents of model i.e. wrong mapping of real world concepts or specified requirements into the model.

To support defect detection we adopted HAZOP, a method from safety-critical system domain originally used to analyze system hazards. Some of its concepts were applied in a reading technique for inspecting UML models. This new technique was named UML-HAZOP. In the sequel we introduce this technique and evaluate its effectiveness and efficiency in comparison to some other reading techniques applicable to UML models.

2. UML-HAZOP Reading Technique

To present UML-HAZOP we need to explain the concepts of the original HAZOP first. Then the presentation of UML-HAZOP is followed by the report from its empirical validation.

2.1 Fundamental Concepts of HAZOP

HAZOP (Hazard and Operability Study) is widely used in the safety-critical systems domain to identify potential hazardous events or states that, if occur, can result in dangerous consequences. HAZOP was invented and used in chemical industry and later was adapted for safety-related analyses in software domain [28]. HAZOP has a precisely defined structure and process (in a way resembling Fagan's inspection). We will however focus here solely on features "borrowed" from HAZOP to develop a reading technique for inspection of UML models.

The technique is focusing on system models and supports their systematic analysis. Potential categories of defects are defined for particular model elements and documented in so called *HAZOP tables*. HAZOP tables are constructed by systematic application of generic *HAZOP guidewords* to the elements and their attributes. Guidewords help to identify potentially dangerous deviations in the state or behavior of the given element. An example set of HAZOP guidewords is given in Table 1.

As an example take statechart model and its element: transition between states characterized by a triggering event. Exemplary applications of guidewords could be:

- event + LATE : "event takes place after it was expected"
- event + OTHER THAN : "an unexpected event occurs instead of the anticipated event"

Not all pairs of guidewords and elements/attributes necessarily make sense. Those which suggest meaningful dangerous situation are included in the HAZOP table dedicated to a given element. Systematic application of guidewords has the advantage of identifying all possible "failure modes" of any element, depending on its semantics.

Table 1. Generic HAZOP guidewords

Guideword	Generic interpretation
NO	The complete negation of the design intention. No part of the intention is achieved and nothing else happens.
MORE	A quantitative increase.
LESS	A quantitative decrease.
AS WELL AS	All the design intention is achieved together with additions.
PART OF	Only some of the design intention is achieved.
REVERSE	The logical opposite of the intention is achieved.
OTHER THAN	Complete substitution, where no part of the original intention is achieved but something quite different happens.
EARLY	Something happens earlier than expected relative to clock time.
LATE	Something happens later than expected relative to clock time.
BEFORE	Something happens before it is expected, relating to order or sequence.
AFTER	Something happens after it is expected, relating to order or sequence.

When it comes to the analysis of a concrete model, each component type present in the model is linked with its copy of HAZOP table. The analyst follows the entries of the HAZOP table which suggest potential deviations of the component in the context of the whole system. If such deviation is assessed credible, the analysis focuses on revealing causes of such situation, consequences for other components and the whole system, countermeasures to apply etc.

2.2 Development of UML-HAZOP

Successful application of HAZOP to software domain resulted in attempts to use it also for other purposes [12], [31] which proved to be a good idea. The HAZOP's basic ideas:

- use of guidewords to identify incorrect states of elements of given type and document them in HAZOP table
- systematic review of whole model with help of HAZOP tables

have confirmed their applicability in various contexts. Using guidewords to identify incorrect states requires interpreting the meaning of the pair: guideword-element, and it is possible to adopt HAZOP to an arbitrary analytical objective just by changing the interpretation.

We constructed HAZOP tables for UML modeling elements by applying guidewords and interpreting them with the aim of detecting design defects just as every inspection technique does. Example of such HAZOP table is given in Table 2. It can be considered as an alternative representation of a checklist. Some differences however exist: this representation forces an inspector to work element by element considering all its possible deviations whereas 'ordinary' checklists often structure the process according to the deviation type, for instance: "Check if multiplicities of (all) associations are correct". Also, working with HAZOP tables requires making explicit decisions about whether each suggestion of defect takes place for a given element or not.

To our knowledge HAZOP has not been applied for the purpose of design defects detection and no related work can be reported in this area. The closest approach to our proposal we are aware of is the method of inspecting requirements scenarios [20]. The method uses a number of very detailed forms to be filled by inspector. Each field of the form has a small scope and focuses inspector on the particular issue, not taking into account other details. An explicit decision of inspector is required for each field of the form. Another related work are the techniques used in industrial practice by Japanese division of IBM – Risk-based design reviews [22]. These techniques, like UML-HAZOP, originated from safety and risk identification methods and currently are commonly used in quality assurance process. Also, a mechanism of generic failure modes (idea similar to HAZOP guidewords) is used by them to support reviewer's work..

Table 2. HAZOP table for element Association

ASSOCIATION		
Attribute	**Guideword**	**Interpretation**
Association	AS WELL AS	Considered classes are not really associated. The association is wrong and should be removed from the diagram.
	OTHER THAN	Wrong type of the relationship. A relationship of another type e.g. *Generalization* should replace the *Association.*
	PART OF	*Association* is correct, but some additional relationship or relationships should be introduced between the considered classes.
Association.Name	NO	The association has no name though it should be named.
	OTHER THAN	The name of association is wrong and does not represent the meaning of the association. The name should be changed or completely removed.
AssociationEnd.Multiplicity	MORE	The marked multiplicity of association is too high (it can concern upper or lower bound of multiplicity).
	LESS	The marked multiplicity of association is too low (it can concern upper or lower bound of multiplicity).
AssociationEnd.Aggregation	NO	The aggregation that should be present is not marked in the diagram.
	AS WELL AS	The aggregation is wrong and should be removed.
	OTHER THAN	Wrong type of aggregation is marked i.e. strong instead of weak or inversely.
	MORE	Too many classes are aggregated; some of them shouldn't be included in aggregation.
	LESS	The aggregation doesn't include some classes (present on the diagram or not) that should be aggregated.
AssociationEnd.Name	NO	The side of the association has no name though it should be named.
	OTHER THAN	The name of association end is wrong or superfluous.

The checklists comprise the main asset of our UML-HAZOP inspection method. The other components include the *process of inspection* and the *supporting internet tool*. The process, controlled by an inspection leader, is similar to classical inspection process with exclusion of the logging meeting; defects found by inspectors are passed directly to author of the model [8]. The tool imports UML models from the CASE tool, generates HAZOP tables, enables

inspector to register results in HAZOP tables and the author to verify them [10]. Recently a new version of supporting tool has been implemented in a new technology and its testing is now in progress.

2.3 Validation of UML-HAZOP

UML-HAZOP was first validated in four case studies during which the technique was applied to UML models developed in real software projects, including two models of systems developed by a major Polish software house. In each case study the technique was able to detect nontrivial defects, despite the fact that the models undergone standard industrial quality assurance practices [8].

In addition to the industrial case studies, to learn more about UML-HAZOP characteristics, we have conducted two controlled experiments. The first experiment's goal was to compare UML-HAZOP with ad hoc reading. Comparison was with respect to effectiveness (percentage of all present defects found) and efficiency (number of defects found per unit of time) of UML class diagram inspection [13]. The experiment took place in October 2003 with the help of 15 last year students of Informatics. The results demonstrated that although UML-HAZOP performed significantly better within its scope, the scope was to narrow and many significant defects went undetected. Defect logs and debriefing questionnaires also shown some problems with learning the concept and use of the UML-HAZOP technique.

The results of the experiment were used to improve our technique. The scope was broadened to cover all categories of defects encountered in class diagrams. Also the structure of HAZOP tables was improved to provide more guidance to inspectors and the training program for inspectors was revised. Then we decided to conduct the second experiment to assess how these modifications affected the technique's potential [9].

The second experiment took place in December 2004, with participation of 17 last year students of Informatics. The same model has been subjected to inspection and all other significant variables were kept as close to previous experiment as possible (to allow comparison between results of two experiments). All students were asked to apply UML-HAZOP. The results obtained were slightly better that for the ad hoc approach. However, the time for inspection turned out to be to short (only 65% HAZOP tables were filled by an average inspector). New HAZOP tables proved to have an adequate scope covering all defect categories.

3. Experiment Design

Based on the validation results as described in section 2.3 and some new ideas about improvement of the UML-HAZOP technique, we have planned for a new experiment.

3.1 Goals and Variables

The research on UML-HAZOP reading technique brought a number of new questions that we wanted to investigate empirically:

- Would the technique prove more effective/efficient if it is modified to be more scenario-oriented like DBR or PBR?
- Will the extended, more comprehensive training program of UML-HAZOP be associated with the improvement in effectiveness/efficiency?
- What is the increment of defects found in the function of time for particular reading techniques (i.e. do inspectors detect defects in a steady pace or e.g. more at the initial period of inspection)?

To answer those questions we designed a new experiment, also in academic environment and similar to previous, but with a significantly larger number of subjects. The experiment had two independent variables:

- **IV1**: Reading technique assigned to subject.
- **IV2**: Time available for inspector to conduct inspection.

The dependent variables included:

- **DV1**: Number of defects found individually by each inspector.
- **DV2**: Time actually spent by inspector checking defects.
- **DV3**: The exact timing of reporting each defect by inspector.

<table>
<tr><th rowspan="2">Group A
Ad hoc</th><th rowspan="2">Group B
UML HAZOP</th><th rowspan="2">Group C
Scenario</th><th colspan="2">Group D
UML-HAZOP, reduced time</th></tr>
<tr><th>Sub group 1
relation-ships</th><th>Sub group 2
classes</th></tr>
<tr><td>Reading
(1 hour)</td><td>Reading
(1 hour)</td><td>Reading
(1 hour)</td><td>Reading
(1 hour)</td><td>Reading
(1 hour)</td></tr>
<tr><td rowspan="3">Checking (2 hours)</td><td rowspan="3">Checking (2 hours)</td><td>Drawing
(0,5 hour)</td><td rowspan="2">Checking
(1 hour)</td><td rowspan="2">Checking (1,5 hour)</td></tr>
<tr><td rowspan="2">Checking (1,5 hour)</td></tr>
<tr><td></td><td></td></tr>
</table>

Fig. 1. Inspection processes defined for each of experimental groups

The following reading techniques were used in experiment:

- **Ad hoc approach** – no guidance focusing inspector's attention.
- **UML-HAZOP** – each inspector used a set of HAZOP tables, either related to classes or to relationships (filling tables from both sets would certainly exceed time available).
- **Scenario-based approach** – after reading source documentation, subjects were asked to draw their own class diagram on the basis of requirements from source documentation. After that they received the inspected diagram and were asked to mark differences between it and diagram they created. Finally, they were supposed to fill HAZOP tables from one of two sets like in UML-HAZOP approach with special attention focused on spotted differences between diagrams.

The second independent variable (**IV2**) is the time that was made available to inspector for performing inspection. We formed two groups using UML-HAZOP technique, however one group (D) had a shorter period of time to complete the task in order to check how does effectiveness change.

For each group two phases were defined: reading source documentation and checking target documentation (UML diagram) with reporting defects. Group C had an additional phase between reading and checking, during which subjects drew their versions of diagrams. In case of group D checking phase time was different for subgroup inspecting classes and subgroup inspecting relationships. All inspectors were working individually. The structure of the inspection process is shown in Fig. 1.

3.2 Subjects and Instrumentation

Subjects of the experiment were 57 students from the last year of Informatics (44 persons) or Telecommunication (13 persons). None of them had previous experience with inspections. Informatics students had a good background in modeling and UML, while Telecommunication students knew only the basics. Being aware of these differences we took it into account when assigning subjects to groups and paid special attention when analyzing results to prevent biasing them by this factor. The students were asked to participate in experiment as a part of the university course.

We followed the guidance related to ethical issues of experiments in academic environment [21] by providing educational benefit to students (theoretical and practical course of software inspections). Also, when grading students we did not "punish" anyone for poor inspection results – the simple fact of participation was the basis of the grade. We believe that this also makes results more reliable – inspections were conducted in conditions more similar to natural ones and without additional pressure.

The object of inspection was the documentation of a billing telecom system which was designed and implemented during a project realized in university. It was the same documentation as in the two previous experiments. The documentation consisted of 16 pages in English and included a diagram (16 classes and 16 relationships) and 15 pages of description of its contents - a report generated from a CASE tool. Subjects were told to search for defects only in the diagram, not in the descriptions. The diagram was known to contain 79 defects (all types, major and minor). The source documentation was an excerpt from system requirements specification and had the size of 21 pages in English.

Two software tools were used in the experiment; first of them was our UML-HAZOP supporting tool described in section 2 (used by groups B, C and D), the other was Mantis Bugtracker, an open source system for reporting errors and change management (used by group A).

4. Conducting the Experiment

The experiment took place in December 2005 and January 2006. Before the experiment itself started, training sessions were organized for subjects to familiarize them with reading technique assigned and software tool used to report defects. Each training session consisted of two parts. The first part was the presentation explaining the experiment design and the role of subjects as well as explaining the reading technique assigned. The second part involved a training of inspecting a small exemplary UML model using the same reading technique as later in the experiment. They also had an opportunity to learn the functionality of software tools they later used during inspections. The assumption was that the training is in the same environment as during the later experiment. This included written instructions to be followed, format of materials handed out to subjects, division of inspection into phases etc.. The difference was in size and complexity of the inspected model and its source documentation and in the time spent on inspection.

The experiment was conduct in five sessions. Table 3 shows how the subject were allocated to groups and sessions.

Table 3. Numbers of subjects in groups and sessions

	Group A Ad hoc	**Group B** UML HAZOP	**Group C** Scenario	**Group D** UML-HAZOP, reduced time	
				Sub group 1 relation-ships	**Sub group 2** classes
Session 1 (10 Dec)	4	5	7		
Session 2 (12 Dec)	8	8	2		
Session 3 (13 Dec)	5	4	5		
Session 4 (9 Jan)				4	
Session 5 (10 Jan)					5
Total	**17**	**17**	**14**	**4**	**5**

Sessions 1-3 of the experiment were conducted in two laboratory rooms. Each room had a supervisor, who was responsible for respecting the time constraints of inspection process, and preventing plagiarism and "group work". Supervisors also registered exact, not nominal time spent by each subject working on each phase.

During the experiment the subjects received the following printed materials:

- instruction containing short description of experiment, its phases, time constraints, login credentials to access software tools;
- source documentation;
- target documentation (model to be inspected) – this document was not available to subjects in reading phase, but given with the beginning of checking phase;

All materials were collected from subjects after the experiment before they left to prevent the "history" threat to experiment validity.

Defects found by inspectors in checking phase were continuously registered in assigned software tool (UML-HAZOP tool or Mantis Bugtracker). Immediately after the experiment, the accounts used by subjects were disabled.

The data collected from the experiment included defects and time of their registration (stored in databases of software tools) and precise time spent by each inspector (registered by supervisor).

There were no serious problems or disrupting factors reported during the experiment. The only problem encountered was with UML-HAZOP tool, which had several moments (2-3 per session) when it became unavailable or responding very slowly for a few minutes and once (for part of group C) it suffered failure and subject had to used file defect logs prepared as a backup.

5. Results

All reported defects were analyzed to assess their validity. We looked closely at the result of each inspector to determine whether he/she shows signs of not conforming to the assigned procedure or a deliberate lack of effort. We also excluded all results below 4 confirmed defects, regardless of a group assigned, assuming that this means either lack of commitment or lack of task understanding. In total, 5 subjects were excluded. For the remaining, the confirmed defects were used to calculate effectiveness and efficiency metrics for each group. Data concerning precise timing of reporting defects was extracted from software tools' databases and visualized in graphs.

The UML model used as experiment object contained quite a few defects, mainly introduced during its design (in addition, some 15 defects were seeded). 51 defects out of total 79 were related to classes and the remaining 28 to relationships. As our objective was to determine the performance of the reading techniques some of which deliberately were focusing on classes or relationships respectively, we applied correcting weights to experimental data in order not to be biased by unbalanced defects distribution. Therefore, when calculating metrics we used "normalized number of defects" – numbers of defects related to classes remained unchanged while numbers of defects related to relationships were multiplied by 51/28.

Although the experiment was limited to the individual phase of inspection, we calculated the performance of pairs – we checked the (average) overlap of defects found by two inspectors. For group A, we considered all possible groups of two inspectors. For groups B, C and D, we considered groups where one inspector was using HAZOP tables dedicated to classes and the other to relationships. We found that the overlap is 17% for group A and maximum of 2% for groups B, C and D.

We defined the following metrics:

- Effectiveness - the normalized number of defects found divided by the number of all known defects present in the diagram,
- Efficiency - the normalized number of defects found divided by the checking time (number of defects per checking hour).

To provide for the same coverage while calculation the metrics for different techniques, we calculated the metrics for pairs of inspectors. For groups B, C and D, each pair combined the focus on classes with the focus on relationships. For group A, each pair combined two randomly chosen inspectors (because in this case the focus of each inspector was the same: all defects).

Mean values of effectiveness and efficiency are presented in Fig. 2 and 3. We can see that all four techniques have quite similar effectiveness. More differences can be noticed in efficiency. Group D working under more rigorous time constraints achieved the best result. Group C performance is about the level achieved by group A (ad hoc) and inferior compared to B and D (which used 'standard' UML-HAZOP). Interesting point is that group C was the most effective among all groups in detecting defects of relationships.

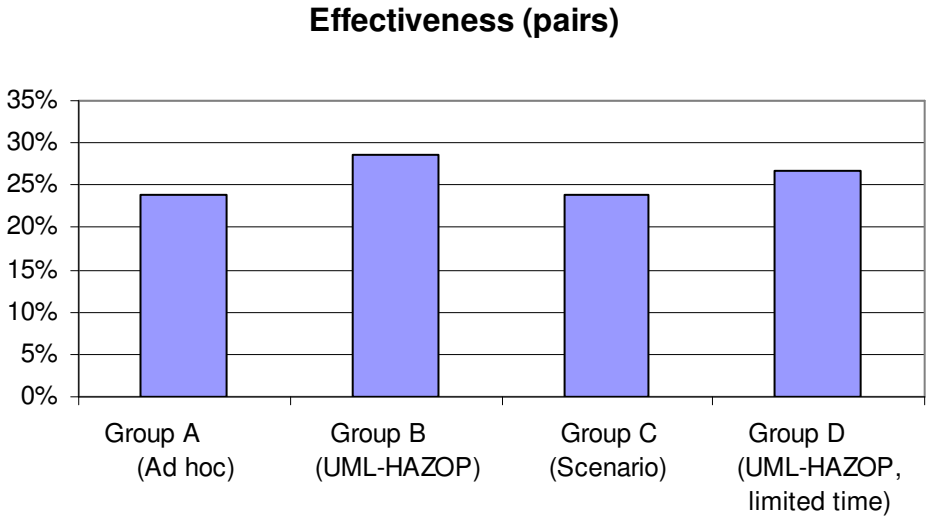

Fig. 2. Comparison of pair effectiveness

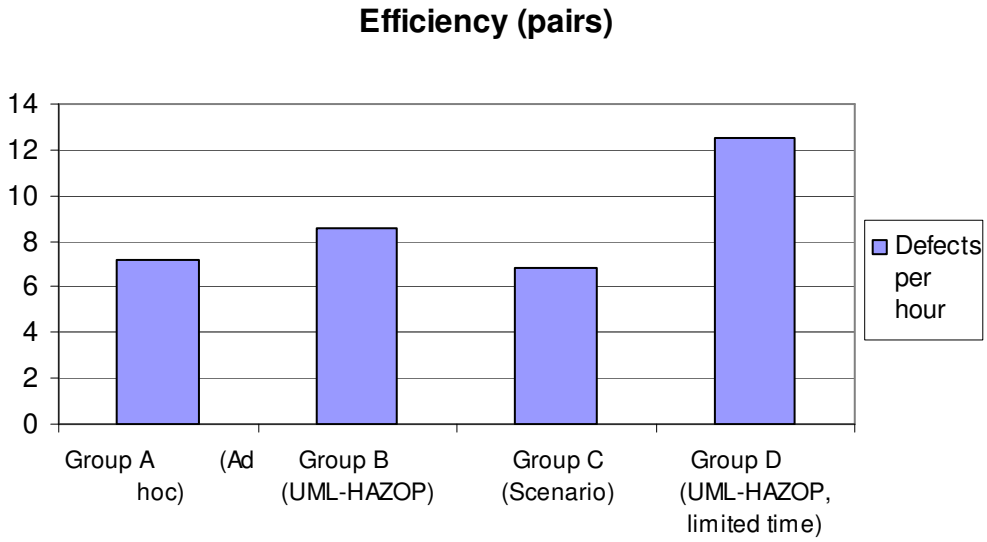

Fig. 3. Comparison of pair efficiency

We also wanted to find out how does the number of reported defects change as the inspection progresses. The hypothesis was that for more formalized reading technique which require filling in HAZOP tables the curve is more linear than for an ad hoc approach where an inspector at the beginning finds a large number of defects and then the rate of finding defects decreases. Fig. 4 visualizes how does the overall number of defects reported in groups A and B change. It can be seen that initially ad hoc approach gets an advantage, but as the checking progresses the performance of UML-HAZOP improves and the lines meet about 100^{th} minute of checking. Presented graph includes only confirmed defects (without false positives), but the curves for all reported defects (not shown here) do not have much different shape and a similar trend is visible for them.

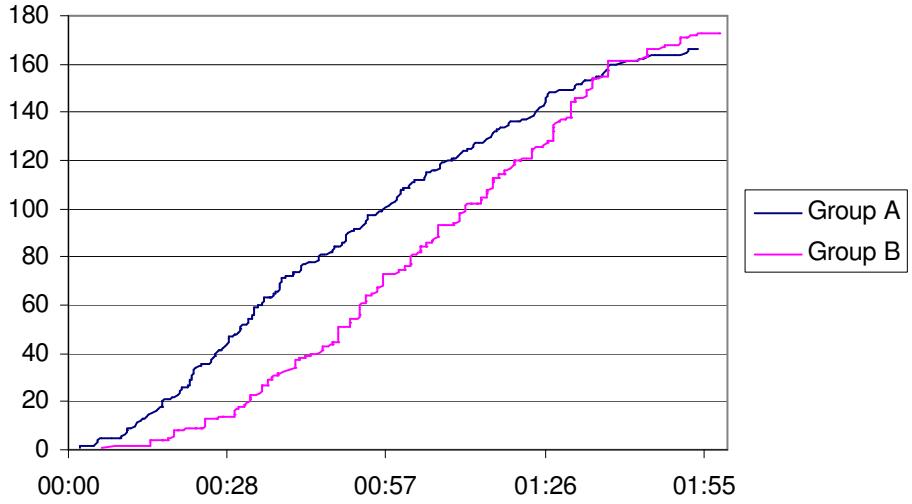

Fig. 4. Number of defects reported as function of time of checking phase

6. Validity Threats

Discussion of validity includes internal and external threats.

6.1 Internal Threats

History effect has to be taken into account as the experiment consisted of several sessions with interval periods and it is possible that subjects discussed the experiment between themselves. Our countermeasure was that they were told not to share information related to the reading techniques, models and defects, and we did not allow them to take any experimental materials away.

Maturation effect is automatically excluded as each subject conducted only one inspection. No changes in data collecting and analyzing were introduced which eliminates the instrumentation threat. Lack of conformance to the assigned technique was detected during analysis of results and the results of such subject were excluded. Selection can be considered a threat as we were limited by availability of students (they chose session they can attend). We assigned subjects to experimental groups in a random manner but ensuring that students of each specialization were distributed equally between the groups.

6.2 External Threats

There are two potential threats preventing generalization of the results – subjects and object of inspection. Subjects being last year students, most of whom are already working in the industry could be considered as close to software professionals. The model subjected to inspection is not necessarily equivalent to industrial products – this is the threat we have to accept.

7. Conclusions

By this experiment we were able get answers to a number of questions. It was found out that extending UML-HAZOP with a more scenario-based approach does not increase effectiveness and efficiency significantly. The effectiveness gain of UML-HAZOP comparing to ad hoc inspection is not significant either. The real added value of using UML-HAZOP is in efficiency which may be further amplified by proper management of the inspection process (assigning adequate time constraints) - as is shown for group D in Figure 2. We can also observe that the dynamic growth of number of all reported defects by group in the function of time is more linear for UML-HAZOP than ad hoc, which first grows rapidly but later slows down.

The UML model subjected to the inspections was not the ‘real’ industrial model, but nevertheless it has been developed in a non-trivial university project. We used in the same model in all experiments to keep the dependent variables constant (the constraints of the experiments). The density and distribution of defects included in the

model also may have influenced experiment results and this suggests that a replication of experiment using other target documentation for inspection would be useful.

For further research we also plan to investigate the maturation effect on the effectiveness and efficiency of UML-HAZOP inspections.

Acknowledgements

We would like to recognize the effort of our colleagues from Department of Software Engineering: Łukasz Cyra, Janusz Czaja, Grzegorz Gołaszewski, Jakub Miler, Marcin Olszewski and Marek Zagórski for their contribution in preparing and supervising the experiment. We wish to thank all 5th year students who took part in the experiment for their time and effort.

We also would like to thank anonymous reviewers for their contribution concerning improvement of this paper and future research directions.

This research was partially supported by 3 T11C 012 27 grant of Polish Ministry of Science.

References

[1] V R Basili, S Green, O Laitenberger, F Lanubile, F Shull, S Sørumgård and M V Zelkowitz, The Empirical Investigation of Perspective-Based Reading, Empirical Software Engineering, Vol. 1, No. 2, 1996, pp. 133-164.

[2] D B Bisant and J R Lyle, A two-person inspection method to improve programming productivity. IEEE Trans. on Software Engineering, Vol. 15, No. 10, 1989, pp. 1294-1304.

[3] M Ciolkowski, C Differding, O Laitenberger and J Munch, Empirical Investigation of Perspective-based Reading: A Replicated Experiment, ISERN Report 97-13, 1997.

[4] R Conradi, P Mohagheghi, T Arif, L C Hegde, G A Bunde and A Pedersen, Object-Oriented Reading Techniques for Inspection of UML Models - An Industrial Experiment, In: European Conference on Object-Oriented Programming ECOOP'03. Springer-Verlag, 2003, pp. 483-501.

[5] A Dunsmore, M Roper and M Wood, Practical Code Inspection Techniques for Object-Oriented Systems: An Experimental Comparison, IEEE Software, Vol. 20, No. 4, July-August 2003, pp. 21-29.

[6] M Fagan, Design and Code Inspections to Reduce Errors in Program Development, IBM Systems Journal Vol. 15 No. 3, 1976, pp. 182-211.

[7] T Gilb and D Graham, Software Inspections, Addison-Wesley, 1993.

[8] J Górski and A Jarzębowicz, Detecting defects in object-oriented diagrams using UML-HAZOP, Foundations of Computing and Decision Sciences, Vol. 27 (2002), No. 4, pp. 197-210.

[9] J Górski and A Jarzębowicz, Development and validation of a HAZOP-based inspection of UML models, In: Proc. 3rd World Congress for Software Quality, 2005, pp. 345-354.

[10] J Górski, A Jarzębowicz, R Leszczyna, J Miler and M Olszewski, Tool support for detecting defects in object-oriented models, Proc. of 10th International Multi-Conference on Advanced Computer Systems, 2003, pp. 103-112.

[11] B Hnatkowska, Z Huzar and L Tuzinkiewicz., Refinement of UML collaborations. International Journal of Applied Mathematics and Computer Science AMCS Vol. 16, No.1, 2006, pp.155-164.

[12] A Hussey, HAZOP analysis of formal models of safety-critical software, in: Proceedings of SAFECOMP 2000, LNCS 1943, 2000, pp. 371-381.

[13] A Jarzębowicz and J Górski, Experimental comparison of UML-HAZOP inspection and non-structured review, Found. of Computing and Decision Sciences, Vol. 30. No. 1., 2005, pp. 29-38.

[14] J C Knight and E A Myers, An improved inspection technique. Communications of the ACM, Vol. 36, No. 11, November 1993, pp. 51-61.

[15] O Laitenberger, C Atkinson, Generalizing perspective-based inspection to handle object-oriented development artifacts, Proc. of the 21st International Conference on Software Engineering, 1998, pp. 494–503.

[16] O Laitenberger, C Atkinson, M Schlich and K El Emam, An experimental comparison of reading techniques for defect detection in UML design documents, Journal of Systems and Software archive Vol. 53, No. 2 (August 2000), pp. 183 – 204.

[17] O Laitenberger, K El Eman, and T Harbich, An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective-based Reading of Code Documents, Technical Report, ISERN-99-01, 1999.

[18] O Laitenberger, K Kohler and C Atkinson, Architecture-Centric Inspection for the Unified Development Process (UP), ISERN-01-01 Technical Report, 2001.

[19] F Lanubile and G Visaggio, Evaluating Defect Detection Techniques for Software Requirements Inspections, ISERN-00-08 Technical Report, 2000.

[20] J C S P Leite, J H Doorn, G D S Hadad and G N Kaplan, Scenario inspections, Requirements Engineering (2005) 10, 2005, pp. 1-21.

[21] C M Lott and D H Rombach, Repeatable Software Engineering Experiments for Comparing Defect-Detection Techniques, Empirical Software Engineering: An International Journal, Vol. 1, No. 2, 1996, pp. 241-277.

[22] Y Okazaki and T Okazaki, Risk-based design review and code inspection, In: Proc. of 3rd World Congress for Software Quality, Munich, Germany, 2005, pp. 159-168.
[23] Object Management Group, OMG Unified Modeling Language Specification, Version 1.5, March 2003.
[24] Object Management Group, OMG Unified Modeling Language Specification, Version 2.0, July 2005.
[25] D L Parnas and D M Weiss, Active design reviews: principles and practices. In: Proc. of the 8th International Conference on Software Engineering, 1985, pp.215-222.
[26] A A Porter, H Siy, A Mockus and L G Votta, Understanding the Sources of Variation in Software Inspections, ACM Transactions on Software Engineering and Methodology, Vol. 7 No. 1, January 1998, pp. 41-79.
[27] A A Porter, L G Votta and V R Basili, Comparing Detection Methods For Software Requirements Inspections: A Replicated Experiment. IEEE Trans. on Software Engineering, Vol. 21, No. 6, 1995, pp. 563-575.
[28] F Redmill, M Chudleigh and J Catmur, System Safety: HAZOP and Software HAZOP, J. Wiley & Sons, 1999.
[29] G Travassos, F Shull, M Fredericks and V Basili, Detecting defects in object oriented designs: Using reading techniques to increase software quality. In: Proc. of Conference on Object-oriented Programming Systems, Languages & Applications (OOPSLA), 1999, pp. 47-56.
[30] L G Votta, Does every inspection need a meeting? In Proceedings of ACM SIGSOFT '93 Symposium on Foundations of Software Engineering. ACM, 1993, pp. 107-114.
[31] R Winther, O Johnsen and B A Gran, Security assessments of safety critical systems using HAZOPs, Proceedings of Computer Safety, Reliability and Security, 20th International Conference SAFECOMP 2001, Springer LNCS 2187, pp.14-24.
[32] Z Zhang, V Basili, and B Shneiderman, An empirical study of perspective-based usability inspection. Human Factors and Ergonomics Society Annual Meeting, Chicago, October 1998, pp. 1346-1350.

Author Bio

Aleksander Jarzębowicz received MSc from Gdansk University of Technology in 2002. Currently he is a PhD student at Gdansk University of Technology and a member of Information Assurance research group. He worked in EU funded research projects: DRIVE (5^{th} Framework Programme) and PIPS (6^{th} Framework Programme). His research interests include software engineering (especially review and inspection techniques) and safety analysis of systems.

Janusz Górski is a professor and the Head of the Department of Software Engineering at Gdansk University of Technology where he leads the Information Assurance research group. The group is developing innovative approaches to software and information assurance focusing on improvement of risk and trust management. He authored and co-authored 7 books and some 170 research papers. Led several software projects in the areas of process control, telecommunications and information management. Acted as the local principal investigator in several EU funded research projects including Environment SHIP, Copernicus ISAT, Copernicus INTACCOMP, 5th FP DRIVE, 6th FP PIPS, 6th FP ANGEL. Co-operates with a number of leading companies providing consultancy in software engineering, IT security, safety and project management. Through his involvement in the European Workshop on Industrial Computer Systems, Technical Committee 7 (EWICS TC7) contributed to numerous guidelines and pre-standards related to critical applications of computer systems. His present interests include software and system engineering, and systems safety, security and reliability with particular stress on risk and trust management.