Applying software engineering testing techniques to evaluate emergency plans

Bruna Diirr
Graduate Program in Informatics
Universidade Federal do Rio de Janeiro (UFRJ)
brunadiirr@ppgi.ufrj.br

Marcos R. S. Borges
Graduate Program in Informatics
Universidade Federal do Rio de Janeiro (UFRJ)
mborges@ppgi.ufrj.br

ABSTRACT

An emergency plan is an important artifact used throughout emergencies. Therefore, it is crucial that this plan covers the different scenarios faced by emergency team and that the prescribed procedures generate the desired results. To achieve these goals, emergency plans need to be exhaustively tested prior to its adoption. Emergency teams usually use simulation to test plans, but it is an expensive approach, demanding the building of complex environments and tools to get realistic situations. To be ready to go through a simulation exercise, a plan should not have many or critical mistakes, otherwise the simulation exercise would be wasted. We present a paired approach to evaluate emergency plans before simulation exercises, by applying software engineering testing techniques: formal technical review (FTR) and mutation testing. The expected result is a plan more ready to go through simulation, as many of the mistakes are eliminated prior to simulation.

Keywords

Emergency Preparedness, Emergency Plan Evaluation, Formal Technical Review, Mutation Testing.

INTRODUCTION

Emergency management is a cyclic process which comprises pre-disaster (mitigation + preparedness), response and post-disaster phases. During the emergency preparedness, a response plan is devised and generated. This document is a guide containing information about the procedures the response team should perform when dealing with emergency events (Haddow et al, 2011; Khan et al, 2008; Llavador et al, 2006; Myers, 2004).

The construction of an emergency plan starts with the specification of a set of scenarios that the plan should cover. For each scenario is designed a set of procedures aimed to return the situation to a stable condition with minimum losses. One of the goals of an emergency plan evaluation is to determine if the procedures prescribed are the most appropriate and effective to control the situation. Another goal is to ascertain that all scenarios are roofed in the plan, reducing the necessity of an "on-the-fly" design of plans. Finally, there is a not so obvious goal that refers to a standardization of procedures in order to facilitate training and the use of resources. Meet these goals is not a easy task.

The techniques extensively studied in Software Engineering for software testing could help the evaluation of emergency plans, in particular, the first two goals. To evaluate whether the procedures for a given scenario are adequate, we propose the use of the Formal Technical Review (FTR), which uses a "cause-effect" and "what-if" analysis to find errors or anomalies in the prescribed procedures. To determine that an emergency plan can meet a set of possible scenarios, we suggest the Mutation Testing, which proposes the inclusion of errors in artifacts to identify omissions and unessential procedures (Campanha et al, 2010; Limaye, 2009; Pressman, 2006). We claim that the adoption of these two software engineering techniques to emergency planning evaluation can improve the plan and, consequently, make it more ready to simulation exercises, the next step of the planning design. This is particular important in environments with less resources for simulation.

The paper is structured as follows: Section 2 enforces the role of the emergency plan in emergency management cycle; Section 3 describes the use of software testing techniques for quality assurance and proposes their use in emergency domain; Section 4 details a proposal to evaluate emergency plans using FTR approach and mutation testing and presents an example of the proposal use; Finally, Section 6 concludes the paper.

EMERGENCY PREPAREDNESS

The emergency plan is an important artifact created and used throughout the emergency management cycle. It contains information about organization infrastructure, associated risks, necessary resources and standard procedures which should be performed when an emergency situation occurs (Gómez et al, 2012; Haddow et al, 2011; Llavador et al, 2006; Penades et al, 2011). However, it is difficult to ensure that the emergency plan can deal with all possible scenarios and presents all necessary information to deal with them (Gómez et al, 2012; Llavador et al, 2006). Given a plan, can we guarantee it (a) has identified all those involved in an emergency or all the equipment available? (c) has thought about all emergency situations which may occur? (d) has detailed procedures to deal with possible scenarios? All this needs to be evaluated before a real emergency occurs, allowing a prior solution of problems.

A strategy to evaluate the plan before an emergency occurs is simulation. It allows examining how the plan responds to imagined scenarios, enabling the identification of changes in procedures, need of new resources and training (Haddow et al, 2011). However, simulation is a very expensive approach, as it demands the building of complex environments and tools to get realistic situations. Therefore, all efforts should be made to reduce basic mistakes or omissions in the plan.

Before going through a simulation exercise, two main issues should be addressed: missing scenarios and inappropriate or ineffective procedures. The emergency planners try to predict all possible scenarios, but it is difficult for scenarios resulting from unpredicted evolution of the emergency and from ineffective actions. Another common problem is to include procedures that are not necessary for certain scenarios, causing extra work by the emergency response teams. This paper proposes the application of testing techniques, which is used by Software Engineering, to overcome these problems when evaluating an emergency plan.

APPLYING TESTS FOR ARTIFACT QUALITY ASSURANCE

The main goal in product development is to ensure that the final result works properly and provides the solution for customer's needs and expectations. To achieve this goal, it is important to identify possible inconsistencies. In Software Engineering context, the software developing process includes a testing step, which aims to verify if the software behavior is consistent with the expected and to find situations which the software presents errors (Delamaro et al, 2007; Limaye, 2009; Myers et al, 2004; Pressman, 2006). Software testing starts with the test cases generation, consisting of input data and the expected result using it (Campanha et al, 2010; Franzotte & Vergilio, 2006).

It is also possible to explore the potential of group work to ensure product quality. Formal Technical Review (FTR) can be defined as an activity practiced by technical experts who follow formal procedures to ensure that the artifact created is consistent with the specification (Johnson, 1994; Lindell et al, 2007). The FTR process is composed of six sequential phases (Johnson, 1994): (a) **planning**, which includes the team selection, functions allocation and definition of input/output criteria; (b) **orientation**, which includes documents distribution, and explanation of objectives/processes/documents to participants; (c) **individual preparation**, where participants take note of product problems, questions and comments; (d) **review meeting**, where participants discuss about product issues and registers the product acceptance/rejection; (e) **rework**, which solves the problems identified in product; and (f) **verify**, which involves the analysis if problems were solved and creation of metrics.

We believe that it is possible to use software testing techniques for emergency plan evaluation. The goal of emergency plan evaluation is verify if all scenarios are covered by the plan and if the events evolve as planned. We made an analogy between Emergency and Software Engineering, where the emergency plan can be seen as the software which needs to be tested and the different scenarios can be used as mass of data during the tests. Similar to software test cases definition, it is impossible to identify all scenarios which can occur during an emergency. It is argued that, if it is possible to determine that the set of scenarios can reveal documents errors, it ensures that the emergency plan will work properly in these situations.

It is proposed the use of mutation testing (Demillo & Lipton, 1978) to evaluate an emergency plan for a set of scenarios. This technique is used to evaluate how a set of test cases is adequate to test software and is based on the inclusion of small errors in software (so-called mutant) to identify errors. To highlight differences between software and the mutant, the results generated by them are evaluated. If a test case can distinguish the mutant and software by generating different outputs we call it "dead". If the mutant generates the same result as software, it is classified as "equivalent". To highlight possible differences between them, new test cases should be built. The aim of mutation testing is to kill all mutants generated. To evaluate if a set of test cases is sufficient to evaluate the software, the mutation score is used (number of mutants killed / (number of total mutants - number of mutants equivalents)). The higher the mutation score is, the set of test cases is more appropriate (Campanha et al, 2010; Delamaro et al, 2007; Franzotte & Vergilio, 2006; Limaye, 2009; Pressman, 2006).

USING FTR APPROACH AND MUTATION TESTING FOR EMERGENCY PLANS EVALUATION

The proposal for emergency plans evaluation involves the use of mutation testing during FTR phases. The steps of our proposal can be seen in Figure 1, which will be explained in the next paragraphs.

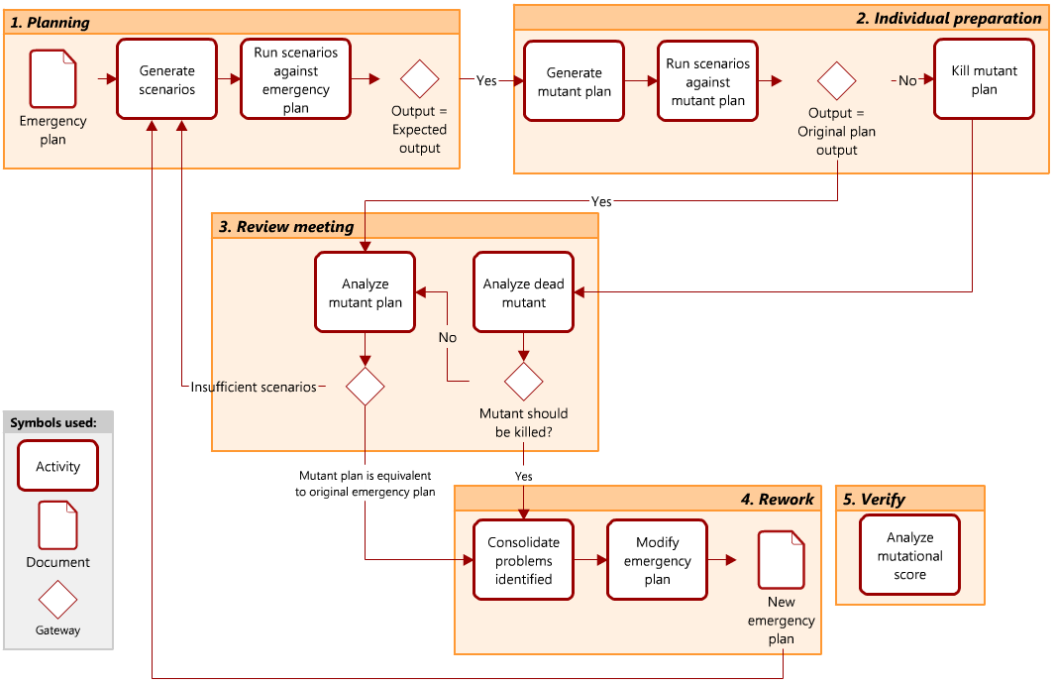

Figure 1. Testing emergency plans through FTR approach and mutation testing

During the planning, an emergency plan is received and the review team generates a set of scenarios, which may be faced during an emergency and that the plan must details procedures to act in such situations, to be used during tests. These scenarios are run against the emergency plan and, if the outputs are according to the expected output, this set of scenarios should be run against the mutant plans. For our example, we can list fire, chemical accident, poisoning and explosion as possible scenarios.

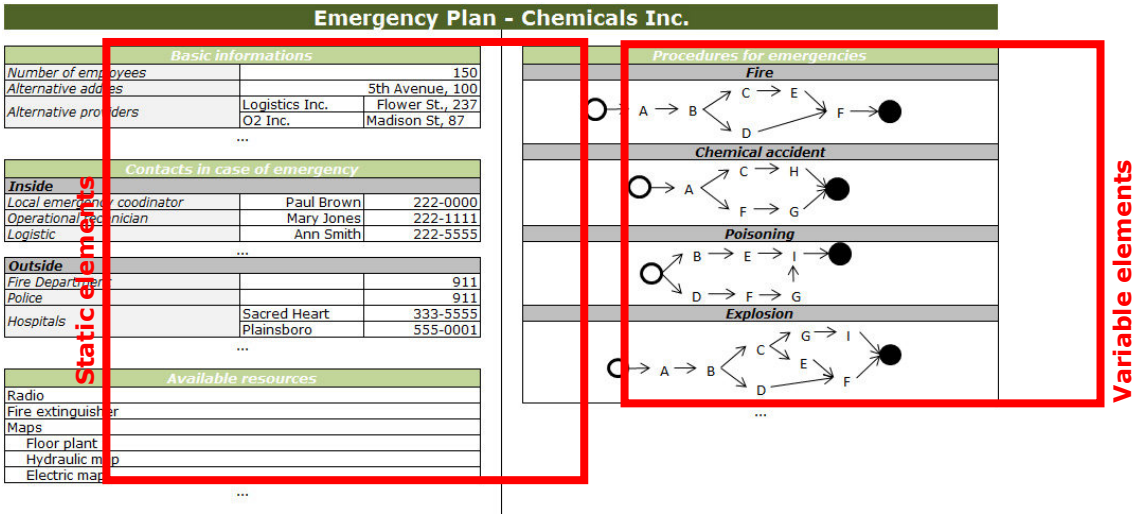

Figure 2. Emergency plan arranged according to static and variable elements

During the individual preparation, the reviewers generate a set of mutant plans by removing parts of the emergency plan. To help this activity, it is proposed a formal organization of the emergency plan in (a) ***static elements***, information which does not vary according to scenario (organization infrastructure, risks which may affect the organization, available resources etc.), and (b) the ***variable elements***, information which vary according to scenario (the response procedures) (Gómez et al, 2012; Haddow et al, 2011; Emergency plans drawn up for different organizations). The removal of static elements evaluates if the scenarios notice changes in the original emergency plan. If they don't, new scenarios that use these elements need to be generated. The removal of variable elements evaluates if there are unnecessary actions being performed or there are missing

actions in a specific scenario. By evaluating these two types of elements, we ensure that the emergency plan presents all relevant information to scenarios and does not present any information which is not used. Figure 2 shows the chemical company emergency plan organized in static and variable elements.

By removing some of these elements, we could generate three mutant plans: "Mutant Plan 1" which doesn't have information about Fire Department (Figure 3a); "Mutant Plan 2" which doesn't have the electrical map (Figure 3b); and "Mutant Plan 3" which doesn't have an activity of the chemical accident procedure (Figure 3c).

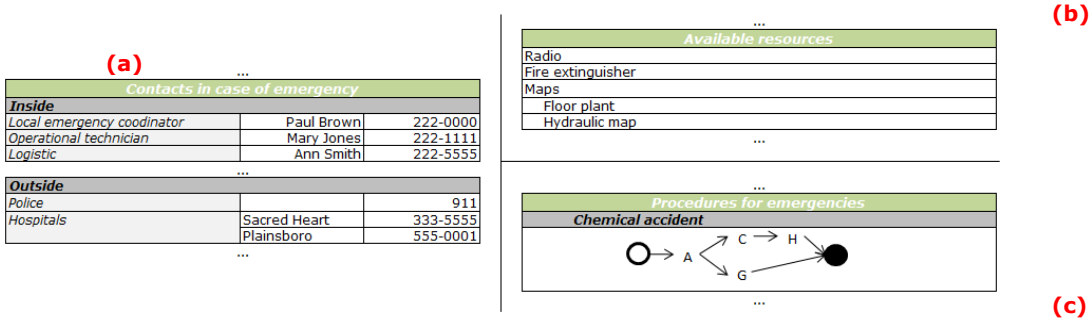

Figure 3. Mutant plans

Reviewers should also run the scenarios against mutant plans and evaluate differences between the outputs generated by mutant plan and the original emergency plan. A mutant plan is called "dead" when the set of scenarios can distinguish it from the original emergency plan by generating different outputs. In "Mutant Plan 1", using the first scenario (fire), the plan cannot be used because it lacks the Fire Department information. Thus, the mutant plan can be considered "dead", since it is possible to distinguish it from the original emergency plan by generating different outputs. Using the same scenarios in "Mutant Plan 2", it was not identified any problem while trying to run the scenarios against the plan. The scenarios were run against "Mutant Plan 3" and, as in "Mutant Plan 2", no problems were shown while trying to run the scenarios against the plan.

During the review meeting, all mutant plans are analyzed. If a mutant plan was killed, the team analyzes if it really should be killed. If a mutant plan was mistakenly killed or could not be killed by scenarios, the team verifies if the mutant plan can be classified as equivalent to the original plan or if it is necessary to generate new scenarios to highlight the differences between plans. We emphasize that new scenarios identification is very useful because, by broadening the range of possible scenarios for a particular emergency, it increases the scope of the emergency plan and reduces the necessity of an on-the-fly design of procedures for unexpected scenarios.

In "Mutant Plan 1" analysis, the team agrees with the reviewer and considers the mutant "dead". In "Mutant Plan 2" analysis, the team concludes that the electric map is a resource which must be used in emergencies that damages the chemical company's electrical installations. However, this scenario (electrical accidents) was not explored during the emergency plan design. It was identified the lack of this scenario only during emergency plan evaluation. In "Mutant Plan 3" analysis, the team needs to understand if the removed activity was disposable or if it is necessary for a scenario which was not previously identified. They conclude that the plan is not wrong because, during the emergency plan design, planners have decided to group all types of chemical accidents in a single procedure. It is recommended to separate the procedure into procedures for each possible chemical accident or to explain that the procedure can be used for different types of chemical accidents, thus some activities will only be used for specific cases.

After the reviewing meeting, all problems identified are sent to emergency planners, responsible for solving inconsistencies (Rework). The revised emergency plan should be sent back to the review team for approval. Also, the scenarios quality is evaluated with the mutation score (Verify phase). It determines if the mass of data (set of scenarios) is sufficient to evaluate an emergency plan and, as a result, the confidence in it. In our example, the original emergency plan should be modified to reflect all issues identified during its review (e.g., the lack of procedures in case of electrical accidents and the lack of explanation that the procedure can be used for different types of chemical accidents). The new emergency plan will be evaluated through the steps described above and it may assist in new inconsistencies identification.

CONCLUSION

The emergency plan is an important tool for decision-making and actions execution during emergencies. However, it is not possible to ensure that this plan can deal with all possible scenarios. Usually simulation exercises is used to test the plan effectiveness, but it is important that this plan should not have many or critical mistakes, otherwise the simulation exercise would be wasted. This paper proposes the use of software testing

techniques in Emergency domain. It is argue that, as these techniques are successful to identify inconsistencies in software artifacts, they could also bring benefits to emergency plans evaluation.

The proposal involves the use of Formal Technical Reviews (FTR) and Mutation Testing to evaluate emergency plans. During the FTR phases, participants generate a set of mutant plans, by removing parts of the original plan, and try to identify errors using the scenarios. The goal of FTR participants is to ensure that the mutation testing was successful, i.e., to ensure that all mutant plans were killed. The combined use of these techniques helps the validation of the set of possible scenarios that the plan can meet and eliminates many of existing mistakes in the plan. It makes the emergency plan more ready to simulation exercises.

It is important to highlight that, despite the benefits of mutation testing to reveal errors in artifacts, its application can be costly. Since it is possible to insert errors in various parts of emergency plan, a great number of mutant plans can be generated, demanding time. Also, as it is a manual testing technique, the quality of the plan as well as the mutants and scenarios depends on the individuals involved in the FTR. Even with these limitations, we believe that these techniques are suitable for emergency plans evaluation.

The application of testing before simulation has been shown to members of the Firefighter Department and Civil Defense in Rio de Janeiro. Although a complete exercise has not yet been performed, the initial reaction of emergency planners was very encouraging. An exercise is being organized and the results should be ready to present in a couple of months. The difficulty of obtaining faster results is the current demand from these two departments during the raining season in Rio de Janeiro.

Although a computer tool is not necessary, its use can help the application of the proposed approach. A prototype is under construction and it will mostly support the memory of the testing procedures allowing the planning team to recall what has been changed in the plan and why.

ACKNOWLEDGMENTS

Bruna Diirr is partially supported by grant 560223/2010-2 from CNPq (Brazil). Marcos R. S. Borges is partially supported by grants 560223/2010-2 and 308003/2011-0 (CNPq) and E-26/103.076/2011 from FAPERJ (Brazil).

REFERENCES

1. Campanha, D.; Souza, S.; Maldonado, J. C. (2010) Mutation testing in procedural and object-oriented paradigms: An evaluation of data structure programs. *In: Congresso Brasileiro de Software: Teoria e Prática (CBSoft)*, Salvador, Brazil, pp. 91-100.
2. Delamaro, M.; Maldonado, J.; Jino, M. (2007) Introdução ao Teste de Software ("Introduction to Software Testing"), Campus, 1st edition (In Portuguese)
3. Demillo, R.; Lipton, R. (1978) Hints on test data selection: help for practicing programmer. *IEEE Computer*, v. 11, n. 4, pp. 34-41.
4. Franzotte, L.; Vergilio, S. R. (2006) Applying Mutation Testing to XML Schemas. *In: International Conference on Software Engineering and Knowledge Engineering (SEKE)*, San Francisco, USA, v. 1. pp. 511-516.
5. Gómez, A.; Penadés, M. C.; Canós, J.; Borges, M. (2012) DPLFW: A Framework for Variable Content Document Generation, *In: 16th International Software Product Line Conference (SPLC)*, Salvador, Brazil.
6. Haddow, G.; Bullock, J.; Coppola, D. (2011) Introduction to Emergency Management, BH-Elsevier, 4th Edition.
7. Johnson, P. (1994) An Instrumented Approach to Improving Software Quality through Formal Technical Review. *In: Proceedings of the 16th International Conference on Software Engineering (ICSE)*, Sorrento, Italy, pp. 113-122.
8. Khan, H.; Vasilescu, L.; Khan, A. (2008) Disaster Management CYCLE – a theoretical approach, *Journal Management & Marketing*, v. 6, pp. 43-50.
9. Limaye, M G (2009) Software Testing: Principles, Techniques and Tools, Tata McGraw-Hill.
10. Lindell, M.K., Prater, C., Perry, R.W. (2007) Emergency Management, John Wiley.
11. Llavador, M.; Letelier, P.; Penadés, M. C.; Canós, J.; Borges, M.; Solís, C. (2006) Precise yet Flexible Specification of Emergency Resolution Procedures. *In: Proceedings of the Third International Conference on Information Systems For Crisis Response and Management (ISCRAM)*, New Jersey, USA, pp. 110-120.
12. Penades, M. C.; Borges, M.; Vivacqua, A.; Canos, J.; Solis, C. (2011) Collaborative Refinement of

Emergency Plans through Public Engagement, *In: International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, Florida, USA, Poster.

13. Pressman, R. (2006) Software Engineering: A practitioner's approach. MacGraw-Hill