Transactions on Information and Communications Technologies vol 11, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

Utilization of FMEA concept in software lifecycle management

N. Banerjee
Isardata GmbH, D-82515 Wolfratshausen, Germany

Abstract

This paper describes how the concept of FMEA, Failure Modes and Effects Analysis, can be utilized to improve the reliability of the software production process resulting in higher product quality as well as in higher productivity. This concept has already been implemented by ISARDATA, a small software company in Germany specialisied in the field of software test and validation, in several software development projects.

The paper begins with introduction of the general principles of FMEA known from applications in various manufacturing industries. The introduction is followed by a brief description of the necessary adaptations of the FMEA method for application in a software production process.

The next section describes the essentials of planning FMEA as an integral part of the software lifecycle management. Since FMEA is primarily the output of teamwork, this section defines practical guidelines for constituting the FMEA team consisting of software developers, testers and quality planners, and for conducting the meetings including defintion of the FMEA objectives of the project.

The following section of the paper describes the main FMEA tasks to be performed by the team. These are the identification of:

a) the structure of the software product in terms of its subsystems, functions, external and internal interfaces and interdependencies;

b) the possible failure modes of the product and their causes;

c) the effects of the failures including calculation of gravity factors;

d) possible measures to prevent and/or correct the failures;

e) test plans to detect such failures during the software development phases;

f) metric for the evaluation of the FMEA results.

The next section of the paper describes how this process can be supported by software tools.

The final section sums up the conclusions.

1 Introduction

Many industrial companies are in the process of implementing TQM (Total Quality Management) with the aim of ensuring, on the one hand, reliable product quality and, on the other, low production costs as well as short delivery times. However, it seems that this ambitious objective can't be achieved without consequent use of preventive quality assurance methods to anticipate potential failures and prevent their occurence. FMEA, Failure Modes and Effects Analysis, is such a method frequently used in practice.

There are two complementary types of FMEA called CONSTRUCTIONAL FMEA and PROCESS FMEA. The constructional FMEA is applied during the starting phase of product development to ensure that while planning and designing a product, foreseeable failures, their consequences, and correctional measures have been taken into consideration. The Process FMEA is applied during planning of product manufacturing process to anticipate failures inherent to production technology and to production management.

A detailed case study of FMEA application in software lifecycle management is not known to the author of this paper. His own experience has shown the necessity of carrying out for practical reasons certain modifications of the FMEA workflow in a software production process.

Modification is required in the set-up of the FMEA team. In the modern software production a software designer has also to be a software developer, and vice versa, as against clearly distinct roles of e.g. product engineer, plant engineer and assembly plant worker in other kinds of industrial production. The FMEA team in software production will therefore have less actors, but several actors will have more than one role to play. This may lead to higher efficiency, because of the smaller size of the team, but also to distorted output if the actors failed to keep a fair balance between their different roles.

Modification is also required in the workflow. It is not practicable to treat constructional FMEA and process FMEA as two tasks to be performed sequentially at different stages of the production, not only because, as mentioned above, exactly the same actors are involved in both tasks, but also because the production of software is an incremental process beginning with the functional specification of the product, when e.g. an integrated software development environment supporting data modelling, user interface design, 4GL or Object Oriented programming is used.

2 Introducing FMEA in software production

For an organisation introducing FMEA in software lifecycle management for the first time, it is extremely important to spend some extra effort at the very beginning of a project to define the goals of FMEA and have them approved by the steering committee of the project, or by other equivalent or even higher authority. Especially the management should be made aware of the following:

FMEA is no substitute for a software process model

A software process model defines the workflow of software production in an organisation. Without such a workflow it is not possible to have a proper coordination between software objects (product) and tools (process) through the different stages of software production. FMEA is an analysis process producing a list of potential failure modes of each product function together with possible correctional measures. This output as such is practically worthless without deployment of the correctional measures, which again depends on the quality of the existing software process model.

FMEA is teamwork

Brain-storming is an essential function of the FMEA process. The pooled know-how of all team members is used to identify potential failure modes as well as the correctional measures. However, the output of brain-storming must be subsequently schematized. The cycle of brain-storming followed by schematization is repeated until for all product functions potential failure modes have been identified, their effects on the functionality of the product have been evaluated, and, wherever deemed necessary, correctional measures have been defined. This is obviously a highly synergetic process whose success depends on strict observation of certain basic rules.

Rule 1 An experienced moderator must conduct the sessions!

Rule 2 The team should never consist of software developers only! User, tester and QA manager are useful, if not essential, team candidates!

Rule 3 Identification of weak spots in a design should never lead to criticism of the author(s) of the design!

Rule 4 The team should remain in office throughout the project!

Rule 5 The FMEA meetings should take place during regular office hours, and not during the spare time of the team members!

The effectivity of FMEA depends on that of existing feedback mechanism

One interesting experience made in FMEA process is that many of the identified potential failure modes are common to a class of software projects. They can be generalised in terms of e.g. application domain of the product, its operational environment, the programming environment chosen for the production, the type of man-machine interaction to be implemented, and so forth. The same is also true of the accompanying correctional measures. It is therefore obvious that by establishing efficient feedback mechanisms in a project, or even better in the organisation, the exploitation of past experience in current FMEA processes will be possible. In this way the productivity of FMEA can be raised very significantly. There are at least three different ways which any modern organisation should be able to use for propagating such feedback:

Common Database containing checklists or even more details on failure modes etc. from past and/or current projects.

Electronic Mail for information exchange between a FMEA team and other experts in the organisation.

Information Seminars conducted by one or more FMEA teams to publish the results of their work to other members of the organisation.

3 The FMEA Process

The following brief descriptions of the FMEA tasks do not consider the process FMEA.

Task: Definition of system model

Several documents are required as initial input for the FMEA process. The most important of them are:

- a user- and application-oriented functional system description;
- a high-level system design description independant of a particular implementation technology;
- a system design description including specification of hardware and software components and their dependencies.

Using these documents a hierarchical system model is created with top-down decomposition of each system component in subsystems until all system functions including hardware components have been described. Complementary to this model, a model describing the possible user-system interactions is also defined.

These two hierarchical models together contain all sources of potential failure modes to be dealt with in the constructional FMEA.

Task: Identification of failure modes

The next FMEA task is to identify potential failure modes for each object of the two hierarchical models. Failure in this context can be defined as any deviation from the expected system reaction. It could be caused either by a design flaw, e.g. an essential function is missing, or by an implementation flaw, e.g. the function has not been properly implemented. This task should identify both types of failure.

Example Let us suppose that one user requirement is a function for drawing graphic objects. In this case, one potential failure mode could be a missing **UNDO** function that would allow user to cancel an action like e.g. deletion/modification of an object. This would be a design flaw. In case the function is there, there could be e.g. one of the following failure modes caused by some kind of implementation flaw:

Failure mode The function can't be activated.

The cause of the failure could be e.g.

- the menu item **UNDO** is not enabled;

Failure mode The function can be activated, but the result is not correct.

The cause of the failure could be e.g.

- the intermediate storage of the object before the last user action is not properly executed;

It is needless to emphasize that the scheduling of the FMEA process should be such that the identification of a design flaw may take place in the early stage of the lifecycle, in order to avoid the necessity of redesign after implementation, which is always a costly affair.

Task: Assessment of the criticalness of each identified failure mode

One basic principle of FMEA is to measure the criticalness of a failure mode, and to define correctional action only if a certain degree of criticalness has been reached. The critical degree, or the risk priority number (RPN), is calculated by multiplying assessment points allocated to

- failure effects (E)
- probability of failure occurence (O)
- probability of failure detection (D).

In case of software it is not always possible to assess O and D. But it is almost always possible to assess either of the two. Therefore, the RPN can be calculated by multiplying E with O or D.

The failure effect E should be assessed using a scale like e.g.

0.0	0.5	1.0
Low	Moderate	Serious

It is, however, necessary to have a common definition of the terms to be used by all team members, as for instance:

Low = The function yields correct results, but handling is cumbersome;

Moderate = Certain options of a function are not executable;

Serious = Unrecoverable loss of data.

Also the probability of failure occurence/detection O/D should be assessed using a scale like e.g.

0.0	0.5	1.0
Unlikely	Evens	Certainty

Example In case of the failure mode **UNDO can't be activated**, the RPN will be high, because the effect (E) will be serious (loss of data) and the probability of occurence/detection (O/D) will also be near certainty. In such a case it is imperative to define a correctional action.

Task: Definition of correctional actions

For each failure mode having a high RPN score a list of correctional actions are defined. Each action should have a reference to the lifecycle stage in which it will be carried out.

Example In case of the failure mode **UNDO function is not enabled** the action could be

- Create a function to enable the menu item following a user action modifying an object (Design/Implementation).

Task: Definition of test cases

Using the available analysis data test cases should be defined describing the test objective, the expected result, and the phase in which the test is to be executed.

Example In case of the failure mode **UNDO function is not enabled**, the test objective will be to verify that the UNDO function is enabled after any user action modifying an object. Therefore several test cases should be defined to verify that the UNDO is enabled after

- modification of an object
- cut, copy or paste of an object
- movement of an object
- execution of the UNDO function (REDO).

Task: Definition of metrics for evaluating the FMEA output

It is quite useful to evaluate the effectivity of FMEA with the help of a few metrics like e.g.

- total number of correctional actions defined;
- percentage of defined correctional actions implemented;
- total number of design corrections defined/implemented;
- Ratio of test cases defined during FMEA to total number of test cases.

4 Software tool support

FMEA is a dynamic process resulting in incremental change of practically all information produced during the meetings. It is also necessary to organise the information in tabular forms to keep track of their interdependencies. Practical experience has shown that the maintenance of dynamically changing structured information is not feasible without use of some kind of computerized documentation system. Several PC-based tools for FMEA documentation are available. They have, however, been designed for FMEA in the manufacturing industry, and not been adapted to the requirements of software production.

As already mentioned above, the productivity of FMEA can be significantly increased if it is supported by

- a common database containing checklists or even more details on failure modes etc. from past and/or current projects

- electronic mail facility for information exchange between a FMEA team and other experts in the organisation.

5 Conclusions

The aim of this paper was to show that the concept of FMEA can be utilized to improve the reliability of the software production process resulting in better product quality as well as in higher productivity. The FMEA itself is no more and no less than a methodically well-organised analysis process. Its results, however, can not be used effectively to improve the product and the production process without the existence of a working software lifecycle-process model. There are at least three aspects of FMEA leading to improvement in product quality :

Definition of a system model The task of defining a hierarchical model of all system functions, which is very often neglected or not done properly in software projects, helps all members of the FMEA team to have a clear understanding of the user requirements.

Identification of potential failure modes This core task of FMEA performed at an early phase of the lifecycle helps to identify inadequate transformation of user requirements to system functions (design flaw), and also potential inadequecies during implementation of the functions (implementation flaw).

Bringing synergy into the process Making expert developers and testers work as a team from the beginning of a project, pooling and documenting their expertise systematically, is perhaps the chief success factor of FMEA, and definitely a very effective way of practising preventive quality assurance.

There are also at least three aspects of FMEA contributing to improvement in productivity through reduction of production cost and shortening of delivery time:

Effective realization of preventive QA The best way to reduce production cost and shorten delivery time is to prevent failures by using the method of "fore-checking".

Definition of correctional actions on a priority basis By using the RPN assessment method, the available resources (man-power, budget and time) can be used well-directed.

Coordination of test planning with system design By coordinated planning of correctional actions and test cases before implementation, the test effectivity can be raised significantly.