Uncertainty Management Framework Elements – Test Based Verification of the Process

DENISS KUMLANDER
Department of Informatics
Tallinn University of Technology
Raja 15
ESTONIA
kumlander@gmail.com http://www.kumlander.eu

Abstract: - Uncertainties management has been recognised already as a crucial part of modern software engineering. Mainly because the weakest element of current best software development techniques is the communication gaps, unclear specification and similar problems leading the process in the wrong direction and therefore producing some extra cost which is not welcome neither by software vendors nor by customers. It is important to handle and monitor such uncertainties on all stages of the project. This part has been described earlier in uncertainty management foundation principles. In this paper we would like to continue describing the framework and introduce the test based verification approach for the software development process. It is based on proactive verification of the process testing it by problems created artificially in order to ensure correctness of work or discover process problems, so we can correct it on early or every stage of the incremental development.

Key-Words: - Software engineering, methodologies, uncertainties management, framework.

1 Introduction
The software development process as any other product development process is a time consuming project demanding sufficient resources and time. Its main goal is to build up a product, which will satisfy customers need, vision and requirements. This should be done in the most feasible way finding a path from task statement to the project completion in the way we spent as less as possible resources avoiding sufficient tradeoffs of project properties like quality. One of the largest problems on that road is the ability to release the right product as we should aim not the requirements and needs of our customers at the start of the project, but at the end of it. Working together with customers we involve co-design [7], iterative development [9, 10, 11] and other techniques [6] in order to find the shortest path and continuously correct the vision of the end product to be in the right position and in the right time with it. At the same time still the customers satisfaction and software projects correctness reviews shows that we are not doing very well since still leaving into the end result a lot of waste [4]. We know that even the most modern approaches to software engineering still do not allow us to be absolutely successful as up to 27% of all projects fail because unsatisfied with the end result customers [1] and only 20% of functionality in average is used “often” or “always” [2]. This is mainly explained by uncertainties, which some authors recognise as different kinds of risks [8] although from our point of view those are very specific and needs a dedicated framework [12]. One example of such uncertainties are gaps between developed software (features, budget) and customer expectations, gaps in the communication [5] between different individuals and unclear visions of the product.

In this paper we are going to expand the uncertainty management framework and see how we can stress test the software development process and proactively discover uncertainty management elements incompleteness or incorrectness disallowing us to deal with such absence of clearness in the right time and on the right stage of the process. We also bring on the table two cases of real live companies suffering from similar problem and will see which approaches worked well been applied there.

2 Potential Process Gaps
In this chapter we will describe the problem we are going to solve in this paper including two study
cases which we will use in the paper to demonstrate how our proposed principals are applied.

2.1 Potential problems
The well-known problem in software development is the inability to predict what should be the end product exactly. Although we know exactly what we are building, it is impossible to predict all influences on the product coming from internal and external sources. Therefore the software development process is now seen as a process of finding the shortest path to the end result by doing increments, employing agile development and so forth including principles like making decisions as late as possible (lean development). This is one kind of uncertainties we have in project. The other typical one is gaps in communication as between team members involved into the project as to external resources or customers.

The software development methodologies deal with such problems differently. Some early ages methodologies assumed that it is possible to deal with uncertainties on early stages building up complete documentation, detailed designed and signing up all documents of the first stage [3]. Unfortunately the right product is not the one which is ordered, but the one which customer would order when it is delivered, so it misses the future vision on the product as it is by nature impossible predict all changes in the business environment that would be exposed during the implementation time frame, although this method works very well in the life critical systems. Besides the signing of specification process of course protects the company by been accused later in building an incorrect product, but still will leave the customer unsatisfied and so will harm the overall relationship as will not lead to win-win situation.

The modern software development techniques like agile or lean development try to eliminate uncertainties on a constant base handling those as errors by implementing short sprints in agile methods [10] in order to review software by stakeholders as soon as possible and consequently accept or reject the results. Unfortunately these techniques still do simplify the uncertainties management treating them too much as errors or simply risk without handling them with a systematic approach. The difference in team members understanding of goals, personal attitude and communication gaps could lead to hiding problems within the development cycles postponing resolutions to the very end and so still producing a lot of design, development and so forth debts making the final iteration to be much longer than expected or revealing problems we tried to avoid by agile approach.

2.2 Study cases

2.2.1 Midsize company
The company is a branch of a transnational corporation and independent at the same time been responsible for a standalone product, which is currently separated from other products of the company. The company has two main locations in different countries, close enough to visit periodically and far enough to be unable to meet together on daily or weekly basis. The company has clearly established departments and separation of tasks making the collaboration not flat and massive. The overall number of employees is close to 50.

The product is a standard massive and complex product, which is sold on several continents and has a stable number of customers. The company aim to release majour versions one or twice a year and updates two or three times a year apart of periodic hotfix releases.

2.2.2 Small highly distributed company
The company operates in the region which is both sufficiently differs from the founders approach to works process organization and has a sufficient lack of personnel. In the result the company is forced to find personnel from other countries located far away. Moreover they do not have any central branch office. Instead each individual actually locates in a separate city. All this make communication between team members, collaboration and cooperation makes the process slightly problematic. Fortunately this problem is insignificant in compare to advantages derived from working with reliable people sharing the same values. At the same time those remaining issues producing uncertainties in the process should be dealt with.

The company produces software and follows the agile process. It has two standard products and a bunch of customer specific projects, some of them they have to support. They have 6 permanent employees and up to 4 working periodically as consultants. They also cooperate with other companies distributing their products as compliments to their products.

3 Proactive Uncertainty Management
The test phase of many development processes traditionally was located after realization
somewhere at the end of the process cycle. The modern software development techniques try to break this habit down, for example by introducing a test driven development or even a test driven modeling of software. Here the process cycle is rotated and the tests formulation is moved upward, so the development phase is considered to be done after all tests are done (as before), but tests are formulated in the beginning as acceptance criteria allowing to move the thinking to the low level already designing the product and think about the variety of cases we should have in mind. A similar question is stated managing uncertainties in the project: should we wait until the end or the middle of the project to realize problems, i.e. be reactive or try to “measure” the process and either prove its reliability or fix problems as early as possible.

3.1 Errors injection
The idea of the errors injection is to try unbalancing the system by injecting problems ourselves into the process and see how the process (the team) will react/recover from this and so either detect leaks of the process or prove its reliability.

The injected problems can have different forms stressing several aspects of the project process one by one or several at once. The following examples are the best selection from our point of view.

- Incorrect specification: the specification, which doesn’t follow the business values of the product or its main goals, doesn’t support standard interfaces and so forth. It is important to ensure that all team members do understand and share product values and targets, are able to differentiate them and follow.

- Incorrect or modified sizing of a task. Here we would monitor how quickly the problem will be discovered and fixed. This also stress test the internal autonomy, which will be described later, as the correcting signal should come from an ordinal team member, not from the leaders as otherwise it is not a self-organised but a hierarchical team.

- Modifying the MDA diagrams making them incomplete or incorrect, leading the team to temporary incorrect decision. It is important to ensure that the team has several stages of verifying models and diagrams and no incorrect decision will dictate future development of the system.

- Injecting errors in the schedule or moving around cards on the board. This is designed to see how quickly the human mistakes of picking up a wrong card or typing into the schedule the wrong card is detected and do we have the correct monitoring system for such cases in place (alternatively for self-organised team: do they monitor activities of each other and rely on communication or each team member activities are isolated and stay within each individual).

- Giving to different team members different or contradictory information. It is important to ensure constantly (i.e. not only once) that team members collaborate, host daily meetings and know what each of them doing, since otherwise the product is not developed in joined direction, there could be tasks done twice or more and so forth.

- Incorrect validation of tasks. We need to see whether we do have a system to record decisions and revise them. A typical mistake that can be seen in many projects is making decisions, which do not follow previously made decision, so the product development goes in rounds making again and again same mistakes. Sometimes teams do know early made decisions, but do not record and so remember reasoning and so tend to take off correct features replacing with some needed for a particular case without having the big picture of the product.

3.2 Formulating a stress scenario for the team
The other technique we proposed is stressing the team with a special task to verify its resistance and correctness of reaction. The aim is to ensure that uncertainties will not produce risks and will not be turn into problems, which will be hard to find in time and so would produce sufficient troubles. The first stress test we could use is adding uncertainties into the requirements and monitoring how the team and the process will cope with it. Potentially, either a product owner / scrum master or team members, during the planning meeting, should question all uncertainties and either clarify them or postpone (reject) the feature until it will be clarified.

The second stress test is to insist as project manager and scrum master on adding more tasks than the team can implement. The aim here is to see whether the team will hide the troubles or trade off with some properties of the product like quality or will communicate the problems open and try to resolve it even admitting that they cannot cope with the task. Unfortunately the inability to admit problems is one of the biggest problems we have seen in projects. Even the agility approach, designed to decrease sufficiently uncertainties, does help if
team members tries to hide problems faking report, error statistics and so forth.

Finally we can question every element of the self-organised teams. As it is well-known the self-organised team concept is based on three types of autonomy they have. The first one is the external autonomy, which is the ability of the team to act independently from the external management correcting them on the constant basis. It means that the team is granted rights to decide by themselves and external influence on it is minimized. The second autonomy is the internal autonomy of the team, which means the team internal procedures of arriving to consensus and defines the ability to cooperate and involve each member of the team into discussions and the decisions making process. Finally there is an individuals' autonomy defining the members' individual freedom to pick up tasks and so be decide how s/he can be most useful for the team instead of doing what is told and nothing more.

If one of the team autonomies is taken away without the team agreement then the motivation of the team and its ability to act rationally within the agile methodology is restricted which lead to additional uncertainties instead of elimination of them. Clearly it is not what project management would like, but what happens sometimes when one or another person miss the agility essences and start to act in his interests rather than interests of the group. This is also often hidden as team members can easily give up own right and become passive ruining the entire strategy of development process. Therefore we would propose giving to the team special tasks revealing their ability to act within earlier described autonomies. The external autonomy can be verified by

4 Lessons Learned: Study Cases
The earlier described ideas were implemented in companies described as test cases in the beginning of the paper. The aim of this was to verify approaches and see which and when can be applied and which should not be used in certain conditions.

4.1 Midsize company
A lot of process elements believed to be in placed has been shown to be not working and consequently corrected by applying earlier mentioned verification methods described earlier. The overall process, existed in the companies, has been generally followed, but had some sufficient leaks, which was constantly producing a lot of headache at each project completion phase. Our collaboration started from the task to find out why the last stage normally takes so much time and why uncertainties the team has to deal at the end is not discovered in time and stay till the end. Practically all proposed techniques worked, although we were not able to prove that the team is truly self-organised. Unfortunately the current situation on IT specialists market doesn't allow to hire only best professionals, although the company has a clear with to do that and so be small but efficient. Some employees were not motivated enough, so instead of self-organised monitoring of uncertainties we have proposed make certain people to be responsible on each stage of the project for monitoring certain uncertainties type in order to close the leaks and avoid hiding problems.

Among other changes done within the organization the most valued by the management team was an established routine of mapping features and specifications to business values of the product. Before, the constant waste of resources was seen when the team implemented functionality either not requested by customers or not very much valued as sometimes reflected a personal opinion of one product owner, which followed his/her momentum wish. In the result up to 25% of resources were saved.

4.2 Small distributed company
This company has shown a true self-organisation and sharing goals and business values of products they developed been stress tested. At the same time a lot of knowledge transfer related issues has been discovered. Sometimes team members didn’t know what others do or did and so a lot of double work has been done before the verification process has been implemented.

We also would like to note that error injection should be implemented in the absolutely controlled manner. Unfortunately the manager of this company was distracted to much with other tasks and in the results some modification went out of control, were left in the model which resulted in extra work at the very end of the projects. In the future every injected error will be registered with a reminder in a dedicated system (in his case MS Outlook) to ensure correct monitoring and control for the verification process.

This company was among first companies were we tested our approach and therefore it was interesting for us to revisit it after some time. Unfortunately we have discovered that not all implemented decisions were followed due some employees’ rotation and the verification process was not repeated. When we re-executed it, we have
discovered that approximately 25% or earlier discovered problems has reappeared and so had to be patched again.

5 Conclusion
In this paper we have expanded earlier described uncertainty management foundational principles with test based verification technique for the software development process. The verification includes error injection and stress tests, which are designed to investigate how the process and involved individuals are able to cope with uncertainties and discover them early in the process. The aim is either to prove that the process follows best practices or do required corrections early to avoid uncertainties leaking through the process making the end product less predictable than we would like to have following modern software development principles and the uncertain management framework. The described verification process has been implemented in several software companies, two of which are described in the paper as study cases. It is important not only implement the verification process, but also closely monitor it in order not to leave artificial problems into the project. Besides, the verification process should be repeated periodically as once done improvements tend to disappear in time especially when new employees come into the organization.

The future work to extend principles described in the system will include building a formal system, which could be potentially automated merging with current standards on monitoring the project status and writing specifications. Besides the classification of uncertainties and their mapping to verification methods is needed to establish a system that can be followed using a check-list.

References: