# Towards Dependable Software: Researching Work-Flow Gaps in the Distributed Organisations

DENISS KUMLANDER
Department of Informatics
Tallinn University of Technology
Raja St.15, 12617 Tallinn
ESTONIA
http://deniss.kumlander.eu

*Abstract:* - Dependable: reliable, secure, having high availability and safety, supporting continuous development concept of software development has become to be very important as an opposite to bad software customers are complaining about. Moving towards dependable software requires understanding of common problems and finding other ways than just testing to produce such reliable software. In this paper work-flow gaps that are specific to distributed organisations are revised. Unwillingness to travel for long distances, communication gaps, lack of information and process monitoring, weak collaboration and teams' internal "wars" for organisational resources are main troubles. Those should be addressed and forecasted rather than afraid or faced. It is the only way to enjoy advantages of the distributed organisations without having too big risk.

*Key-Words:* - Work-flow gaps, distributed organisations, dependable systems

## 1 Introduction

The high price of improperly developed, incorrectly designed and/or inconsistent products is paid nowadays by many frustrated users. Researches shows that nearly one third of all projects fail because customers are not satisfied with the delivered software [2]. A situation with "successful" projects is not much better: only 20% of functionality (in average) is used "often" or "always" and 16% "sometimes". The remaining 64% is either never used or used just occasionally [5]. Bad software is actually not a new trend. Customers are started to complain practically right after software started to appear. During all those years software development process has passed several evolutional changes and lately strictly formalised process of requirements specifications and automated verification systems [3] were aimed to solve all those problems. Unfortunately some important problems remained still open. It happens mainly because testing is not the only way to improve the result and its' cost is much higher than any other possible improvement during projects because testing is done during a relatively late project phase. In recent years, dependability - an integrative concept comprising such criteria and sub-criteria as reliability, security, continuous development, availability, safety [1, 8] - has become to be a very important concept. Moreover the dependability is important for many life-critical systems where many modern concepts cannot be applied. Those are mostly produced for the business sector software and are aimed to maximised functionality and minimise the cost sometimes compromising its quality. This is not acceptable in avionics, vehicle control, submarine software systems and so forth.

In this paper different types of gaps occurring in projects' work-flows are researched to identify major problems that lead to bad software appearance in order to address them properly and enable to move the software development process toward construction of dependable systems.

## 2 Distributed organisations

A distributed organisation in the context of this paper means an organisation that has the following properties:

- It has more than one office;
- Offices are located on a sufficient distance from each other (i.e. those are not located on different floors of the same building, but are located in different towns, countries or even continents);
- All those offices do participate in the core business activity (in our case in the software development process) and none of those can be removed without destroying the process flow. In other words those offices are not

something like sale offices elimination of which means just loosing some sales in certain areas and doesn't affect future release of the main product.

There are much more distributed organisations than it looks like at the first glance. The larger a company become, the more is probability that it will be distributed one although a lot of starters are single-location companies. Below a short list of major reasons while companies do become distributed is presented.

All those reasons can be divided into two groups, where the first one contains cases, when a company is changed to the distributed one by its own wish, and the second group contains cases, when the company had to become distributed because of either outside factors or decisions that were not specifically done due advantages of being distributed (so some other considerations were main drivers converting a single-located company into the distributed one).

- Companies become distributed by their own wish since:
  - The development process will be cheaper. For example an organisation can establish an office in another country, where developers cost per hour is much lower than in the "native" country. Basically we could include into this reason also organisations that are using outsourcing, although principles of how over-sea branches are controlled can vary considerably. It is worse to notice that outsourcing is used nowadays more and more as a result of the following issue as well;
  - A misfit of a skilled personnel location(s) and product markets. It is possible to identify here two subgroups of reasons, which looks very similar initially. The first one is formulated from perspective of a company selling independent products in many countries: a product is not developed in countries where it will be sold as the development is concentrated in dedicated development centres all over the world. The second group of reasons is formulated either from a single product companies perspective (of is a number of products is limited) or from a single market one – the company wish to develop product in one central place, but has to distribute itself since there is no enough skilled developers in the "native" (markets) areas, so other teams are established somewhere else. Notice that here the distribution doesn't happen to decrease the cost of development, but rather to have an opportunity to develop a product using other work/force markets. Moreover the cost of development could even grow, if a team of highly professional specialists is hired.
- Companies become distributed because of external (concerning a decision to become distributed) reasons:
  - A company could become distributed after buying other companies located in other geographical regions and including their products or/and teams into the core activities or products' lines;
  - Company branches have to work together although it wasn't planned so initially. For example, each group was independent some time ago (were building isolated products for isolated markets), but starting from some moment they have to integrate their software;
  - Globalization of operations, i.e. a need to extend business into other countries. This reason often forces an organisation to extend products functionality by anticipating other countries' requirements or establish there teams for bespoken projects. In the last case this isolated team can be included into the main team later (versus - will have to work together with the main team as their features are included into main / standard product);
  - A need to cooperate with partners in other countries, integrate software etc.

This list of reasons demonstrates, from our point of view, that a distributed company is not an artificial, purely theoretical case, but rather is a reality that our world faces nowadays. The number of such companies permanently grows because of globalisation and improvements of e-channels improving branches communication process. On the whole until now we have been only talking about why companies do become distributed, and what are advantages of this decision. The next chapter discusses what drawbacks are.

# 3 Work-flow Gaps

A work-flow gap is a certain trouble that negatively affects the normal flow of the work process. The negative effect can have many faces and some of them are listed below. First of all a gap can corrupt outputs of a certain project stage corrupting the whole work flow. As a result the work flow could include invalid routes; the following stages could produce incorrect results and so forth. Secondly the gap can be so serious that it will not be possible to continue at all or the discussion will go into some kind infinitive loop. Finally a gap can slow down the work process (so it is not corrupted, but the effect is surely negative).

The "certain trouble" part of the definition refers to the fact that there are different types of the gap. For example a communication gap was defined in our earlier articles as a problem in the communication process that makes the transferred information to be either lost or deformed [6]. In this chapter we are going to analyse the distributed organisations case and identify gaps that are specific for it within the software development process.

First of all let us mention a problem that is produced directly by the distance between sites – a lot of workers are not willing to spend their own time out of their homes as business trips normally require. During such trips an employee is usually restricted in his/her private time wasting opportunities, cannot spend evenings with his/her family etc. This affects the normal work flow if such persons are key persons in an organisation and their replacement is not always a way to solve this problem because of shortness of skilled personnel and high competition among companies nowadays. For this reason a company should be looking for more communication ways before those troubles will lead to any sufficient problems.

Secondly, basing on our consultancy experiences, we can claim that a lot of modern software development models are showing ideal results only in ideal environments, when all team members have no restrictions communicating to each other and moving a project forward. Unfortunately the real life is much more complex and the distributed organisations case is one of those. In practise there is a lot of communication gaps that do really affect the work-flow a lot. Of course, some of those gaps are connected to distributed organisations and some of them are not. Anyway communication gaps are gaps, where information, which is send, is corrupted during the transition process and therefore doesn't equal to the received one. There is a list of major communication gaps' types.

- Difference between persons in skills, backgrounds and experiences. It is also possible to include into this group culture differences, which is usually the only communication gap that is mentioned.
- Restrictions on communication like having to phone up, send emails etc. instead of talking face to face. Different articles say that the visual feedback provides from 20% to 40% of information [4, 7]. So, the lack of the visual feedback of an opponent reaction sufficiently restrict opportunities to understand each other and increase probability that information will be received incorrectly and as a result produce a lot of problems. Of course this issue depends a lot on facilities in use – modern technologies make the communication process much more transparent. Unfortunately not all companies do use those technologies and those will anyway not be able to eliminate the "none visual" communication effect completely.

There is another important issue, which is partly connected to the communication gaps that were discussed above. A work flow communication always goes from one person to another synchronously with moving a process (project) from one stage to another. Here under a communication from person to person we mean a process of transferring information. Mostly it is a process of sending outputs of one stage to trigger the next one. If any person that is involved into this chain is weak in getting or sending information then it will corrupt information, outputs and requirements greatly affecting a project's end result. Notice that although this issue exists in all companies the more distributed the company is the more dramatic effect the weak part of the chain will have on the project.

One more typical gap appearing in the distributed organisations is quite a weak monitoring of the situation over an edge connecting distributing offices. A manager cannot be in another location each day and have to overcome a sufficient distance to reach the monitored location. Usually he is using other channels instead of travelling and those are rather restricted in compare to face-to-face communication and possibilities to see everything by his own eyes. In that case the risk of project's failure is growing. Sometimes consequences are not so dramatic, but rather numerous and stable – inability to meet a schedule after finding that developers have not reported their actual work progress, misfit of certain functionalities, knowing

nothing or a little about team members abilities and skills etc.

As it is complex to communicate and monitor teams over a distance the team on another end (in compare to the managers' location) is often organised as a partly self-sufficient. Sometimes such team becomes more and more independent. The main reason of this is a lack of collaboration. As a result of this segmentation process each such team starts to fight for resources against other teams within the organisation forgetting to care about customers. This means that the company starts to produce customer unfriendly product; products that do anticipate just high level requirements, i.e. are having total misfit in details (either wrong realisation, or unusable in practise). Those products are neither reliable nor dependant.

The next problem is similar to the previous monitoring one, but is formulated from the other team point of view, in other words form a perspective of a team that is located far away from the management centre. The distance team usually has also a little information about what is happening on another end. As a result, it makes harder to plan their work properly as there is no info on future plans, releases, specifications etc. The smallest consequence of it will be improper time spent, for example because of re-doing things due new plans although feature could be developed properly knowing all requirements and required enhancements. The more sufficient consequence can be stress from doing an empty work/redoing it, decreased motivation to do the work (as anyway it should be probably redone) and so forth.

The early mentioned need to communicate over a sufficient distance has more effects on the work flow than just certain restrictions on the communication channels, i.e. on information that is sent. The distance slows down the communication a lot. First of all people in many cases need to communicate over emails and this produces messages ping-pong with a slow reaction on each message. Of course emails are useful when you do need time to re-think your answer or collect more information, but it cannot be seen an alternative to a meeting where ad hoc answers are required. Secondly it is very hard to organise meetings and coordinate people activities especially if those are located in different time zones. Besides key persons, teams and ordinal employees are collaborating much less. This can lead to a work (for example a research) that is done twice or more within one organisation as others don't know what others did before. Finally we arrive to the problem that is the most important one from our point of view. It is not possible to force somebody to do anything over such a distance or at least it is really problematic. For example you cannot just walk into a business analyst's office and asks him to have a quick look on the project to discuss stopovers. Notice also that people tend to react lowly on phone calls by either not answering to those or having it without enough respect.

Sometimes companies that are facing all those problems try to establish a highly formal and hierarchical structure of the work- and information flows by specifying strict rules of moving from stage to stage, from department to department (read from person to person). The main danger of this – there is not way to restore missing information if any node of this chain appears to be weak, i.e. if the system is not self-restoring like a system when you can verify results locating close to persons who produced those by having heard something he was talking (may be during informal conversations). Summarising all we said about formalising the process, we could state that: from one hand formalisation allows establishing a system ensuring avoiding some types of gaps, from another hand it practically always means no unofficial contacts, i.e. cuts all alternative communication channels by relying exclusively on the official one, which also could have drawbacks.

# 4 Conclusion

The price customers are paying nowadays for faulty or incomplete software delivered by many software vendors is very high. The number of never used features is very high and is also a sign of bad software. In recent years, dependability - an integrative concept comprising such criteria and sub-criteria as reliability, security, continuous development, availability, safety [1, 8] - has become to be a very important concept of software development. Moving toward dependable software requires understanding of common problems to find ways, others than testing, to produce reliable software. In this paper work-flow gaps that are specific to distributed organisations were reviewed. Unwillingness to travel, communication gaps, lack of information and process monitoring, weak collaboration and teams internal "battles" for organisational resources are main troubles. Those should be addressed and forecasted rather than afraid or faced. It is the only way to enjoy advantages of the distributed organisations without having too big risk.

*References:*

[1] A. Avizienis, J.C. Laprie and B. Randell: Fundamental Concepts of Dependability. *Research Report N01145*, LAAS-CNRS, April 2001.

[2] E. N. Bennatan, K.E. Emam, Software project success and failure, Cutter Consortium, 2005, http://www.cutter.com/press/050824.html

[3] T.T. Dinh-Trong, A Systematic Approach to Testing UML Design Models. Doctoral Symposium, *7th International Conference on the Unified Modeling Language (UML)*, Lisbon, Portugal, 2004, pp. 10-15.

[4] K. Hadelich, H. Branigan, M. Pickering, M. Crocker, Alignment in dialogue: Effects of visual versus verbal-feedback, *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue, Catalog'04*, 2004, pp. 35-40.

[5] A.A. Khan, Tale of two methodologies for web development: heavyweight vs agile*, Postgraduate Minor Research Project*, 2004, pp. 619-690

[6] D. Kumlander, Software design by uncertain requirements, *Proceedings of the IASTED International Conference on Software Engineering*, 2006, pp. 224-2296.

[7] R. Ludlow, F. Panton, *The essence of effective communication,* Prentice Hall, 1995.

[8] B. Meyer, Dependable Software, *Dependable Systems: Software, Computing, Networks*, Lecture Notes in Computer Science, Springer-Verlag, 2006.