Building Dependable Software in Distributed Organizations: Bridging Workflow Gaps

D. Kumlander

***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 inconsistent faulty software customers are complaining about. Moving towards dependable software requires understanding of common problems occurring in nowadays software engineering business despite all modern approaches. Distributed organisations became quite a standard in software business and workflow gaps specific to distributed organisations are revised in this paper. Unwillingness to travel, communication gaps, lack of information and process monitoring – what are some of those problems. The paper proposes also some novel approaches to bridge those gaps.

***Keywords*—** Workflow gaps, distributed organisations, dependable systems.

I. Introduction

UNFORTUNATELY nowadays software engineering reports contain numerous complains of customers frustrated by improperly developed, incorrectly designed and/or user unfriendly products. Nearly one third of all projects fail since customers are not satisfied at all with the delivered software [2]. A situation with projects that were accepted by customers is not nice as well - 64% of delivered software functionalities (in average) is either never used or used just occasionally, 16% "sometimes" and only 20% of functionality is used "often" or "always" [5]. Bad software is 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 including iterative software engineering, agile approaches and strictly formalised process of requirements specifications and automated verification systems [3]. 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. At the same time potential problems do not end with inventing a software engineering methodology, which been applied in an ideal environment, solves them. The real practice is much more complicated and one of such special cases is examined in this paper – distributed organisations.

The second chapter of the paper defines what is meant under distributed organisations, and what is a probability and reasons for an organisation to become distributed. Thereafter different types of gaps occurring in projects' workflows are researched to identify major problems producing inconsistent software in the result. The forth chapter discusses methods to bridge those gaps in order to enable moving the software development process toward dependable one. The final chapter concludes the paper.

II. Distributed organisations

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

1) It has more than one office;
2) 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 rather located in different towns or countries or even continents);
3) All those offices 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.

There is much more distributed organisation than it looks like at the first glance. The larger company become, the larger is probability that it will be distributed although a lot of starters are single-location companies.

Major reasons, why companies are becoming distributed, can be divided into two groups.

1) Companies become distributed by their own wish since:
   a. 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 item organisations using outsourcing as well, although a control over branches can vary dramatically.
   b. A misfit of a skilled personnel location(s) and a product market etc. It is possible to identify here two main subgroups of reasons. The first one is – a centrally developed product is sold in other regions. Business analysts and/or project management is located in the market country to define requirements correctly, i.e. are ordering a product from a central development office. The second case is opposite - distribute an

organisation since there is no enough skilled developers in the "native" (markets) areas, so other teams are established somewhere else. Notice that this distribution doesn't aim to decrease the cost of development.

2) Companies become distributed because of external (to a decision to become distributed) reasons:
    a. A company could become distributed because of buying other companies locating in other geographical regions and including their products or teams into core activities or products lines;
    b. Company daughters or branches have to work together although it wasn't planned so initially. For example, each group was independent some time ago, but now they have to integrate their software.
    c. Globalization of operations, i.e. a need to extent business to other countries. This reason forces an organisation to extend products functionality to other countries requirements or establish there teams for bespoken projects.
    d. A need to cooperate with partners, integrate software etc.

This list of reasons demonstrates that a distributed company is not an artificial, purely theoretical case, but is a reality that our world faces nowadays. The number of such companies permanently grows because of globalisation and improvements of e-channels allowing branches to communicate better and better. At the same time there co-work is not easy and the next chapter discusses what drawbacks the decision to become distributed has.

III. Workflow Gaps

A workflow gap is a certain trouble that either corrupts the normal work process (for example by corrupting outputs of a certain project stage) or makes impossible to continue or sufficiently slow down the process.

There could be different types of gaps. For example the 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 such gaps that are specific for distributed organisations and are related to software engineering activity.

First of all let us mention a problem that is directly produced 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. For example during such trips an employee cannot spend evenings with his/her family. This affects the normal workflow if such persons are key persons in an organisation and persons replacement is not always a way to solve this problem in nowadays shortness of skilled personnel and high competition among companies. Therefore a company should prepare in advance to this situation by finding more ways to communicate.

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 in communicating and moving a project forward. Unfortunately the real life is much more complex and the distributed organisations case in one of those. In practise there is a lot of communication gaps. Some of them are connected to distributed organisations (a sufficient distance between people) 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.

1) Difference between persons in skills, backgrounds and experiences. It is also possible to include into this group cultures differences, which is usually the only communication gap that is mentioned.
2) Restrictions on communication like having to talk via phone, send emails etc. instead of talking face to face. Different articles say that visual feedback provides from 20% to 40% of information [4, 7]. So, lacking of "visual" feedback of an opponent reaction is quite a sufficient restriction, which potentially produces a lot of problems. Of course this issue depends a lot on facilities in use – modern technologies make the communication process more transparent. Unfortunately not all companies do use those technologies and those still cannot eliminate the "none visual" communication effect completely.

A workflow communication always goes from one person to another synchronously with moving a process (project) from one stage to another. If any person involved in 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 can appear in any type of companies the more distributed the company is the more dramatic effect the weak part of the chain will have on the project since communication is weak and there are less ways to restore (or identify) the corruption

One more typical gap appearing in the distributed organisations is a weak monitoring of the situation over an edge connecting distributing offices. A manager cannot be in another location each day and has to travel a sufficient distance to reach the monitored location. Therefore 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 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, having a restricted information about team members abilities and skills etc. mis-assigning tasks to individuals.

As it is complex to communicate and monitor teams over a

Manuscript Received May 15, 2007; Revised received Dec. 21, 2007

distance the team on another end (in compare to the central location) is often organised as a partly self-sufficient. Sometimes such team become more and more independent due a lack of collaboration. Finally, each team starts to fight for resources against other teams within the organisation, forgetting to care about customers. This results in highly customer unfriendly products, products that anticipate just high level requirements having a total misfit in details. Those products are neither reliable nor dependant.

The next problem is similar to the previous monitoring one, but is formulated from the other end (distance) team point of view. The distance team usually has also too little information about what is happening in the centre. It makes much harder to plan their work properly, prioritise and schedule tasks. It is unfortunately quite a often case when the team spends time improperly because of redoing things, reworking architecture etc as more information is available now. The more sufficient consequence can be a 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 earlier mentioned need to communicate over a sufficient distance has more effects on the workflow than just restrictions on the communication channels, i.e. information that is sent. The distance not always corrupts information or cuts it. It is possible that the distance just slows down the communication a lot, so it is not corrupted, but surely affects the workflow negatively. First of all people need to communicate over emails and this produces messages ping-pong with a slow reaction on each message. Secondly it is very hard to organise meetings and coordinate people activities especially if those are located in different time zones. Thirdly key persons, teams and just employees are collaborating much less. This can lead to a work (for example some kind research) done twice or more from the organisational perspective. Finally, it is not possible to force somebody to do something over such a distance. For example you cannot walk into a business analyst's room 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 or getting it without enough respect.

Different time zones produces time planning connected workflow gaps, when one team has to wait for another team to do their tasks, when the other team starts later or is already finished work for today.

IV. Bridging Workflow Gaps

Sometimes companies that are facing all those problems try to establish a highly formal and hierarchical structure of the work- and information flows moving from stage to stage, i.e. from a department to department (read from a person to a person). The main danger of this – there is no way to restore missing information if any node of this chain appears to be weak. The system is not self-restoring like a system when you can verify the result been close to the previous to the previous person and having heard something he was talking. Therefore, although processes formalising allows establishing a system to ensure avoiding some types of gaps, it practically always means no unofficial contacts. So it cuts all other alternative communication channels and relies exclusively on the official one, which is inefficient in distributed organisations as it was already shown.

A. "Ambassadors" driven workflow

The previous chapter has identified a weak communication as one of the main reasons companies fail in the work processes. An idea of functional or positional "ambassadors" can be a "treatment" in this case, i.e. the physical position can be divided over distance as the work is. The main problem of the weak communication is impossibility to get quick answers, restrictions on the full communication and practical unavailability of the other side persons for small details and questions. In this situation there is a clear need to have someone locally who will be able to answer those questions, i.e. to act as a representative of either a person or even a team located on the distributed edge. Our practice has shown that a lead developer unofficially become a representative of designers on the distributed edge, designers become a representative of project management etc. It is possible to make this practice officially making it efficient as the person in charge has more possibilities to demand, ask questions etc. The development should not be driven by believes having no information, but by precise knowledge and definition.

The ambassador idea should not be restricted to delegating responsibilities downward. It is also extremely important to have a delegate of a team in the headquarter, who will be able to provide the team with information, stay for the team interests and cooperate on the high level with other teams avoiding resources fights.

B. Infrastructure

Another gap that was identified earlier is a lack of transparency about project statuses, feedbacks, knowledge (conducted tests and researches), documents etc. This gap definitely needs company management attention and the best solution will be to bring the information out of the physical location avoiding the distributed offices impact as much as possible. Of course the restricted communication will stay, but the information will not be stacked in physical offices. It should be visible to each member of the organisation (accordingly to security rights of course) in the virtual environment. Therefore there is a need to build an infrastructure to store information, keep projects track and collaborate. Unfortunately a lot of organisations are forgetting about the primary goal of building such infrastructure. They build an infrastructure, which is accessible all over the company, but not the collaborative one. The main dangers are:

1) Slow infrastructure;
2) Containing only some information needed, not integrated with other systems;
3) Users unfriendly;
4) (*as a result of previous*) A system, which is not popular in

the organisation, i.e. a system which key persons do not use.

C. Meta-team

The communication over edges is obviously more complex than face-to-face one, therefore it is extremely important to pay attention to all details improving the co-work. First of all it is important to train people to communicate and probably select individuals with corresponding skills hiring personal (i.e. rate communication skills much more than normally). Individuals' properties as "friendly", "open minded" and responsible become crucial.

Thereafter it is important to ensure that communicating workers do know well the language they are using as distributed organisations include usually none-native speakers of the company official language. It doesn't only mean learning the language (which is obviously very important by itself), but should be extended to learning habits, culture etc of other workers' social and national groups.

Finally it is important to build a meta-team in the organisation, i.e. a team of employees working in different places. The efficient communication requires them to know each other, have free, open style of communication hopefully including unofficial contacts. That is the most efficient way to balance communication restrictions within the organisation produced by distances.

D. Communication channels and habits

The current team have to accept communication over e-channels and therefore re-built own communication habits and preferences. It is amazing how do people fear talking over webcams preparing for it, although do not care about talking face to face. The only reason of that is – people used to meet other people in everyday live and treat using web cams etc as something special and therefore something that needs preparation.

It is important to make using e-channels ordinary, habitual and natural and therefore friendly and efficient.

V. 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 reported by different researches 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 that just testing, to produce reliable software. The distributed organisations case is one that appears often in nowadays global world, which produces certain specific troubles. Unwillingness to travel, communication gaps, lack of information and process monitoring, weak collaboration and teams battles for organisational resources are workflow gaps that were reviewed in the paper.

It is important to bridge gaps and proactively react on such dangers rather than afraid them or produce faulty software reacting on occurred risks. It is the only way to enjoy advantages of the distributed organisations without having too much risk. An "ambassador" driven process' workflow allows overcoming distance gaps increasing efficiently and returning the process on track – it is driven by exact knowledge as a single located companies, not by believes, i.e. uncertainties produced by distances. The monitoring, feedbacks and workflow transparency can be achieved by a dedicated processes informational system for internal collaboration. The collaboration is also a cornerstone in communication and therefore it is important that employees do understand each other using none visual communication channels. People should be friendly, open minded and ideally form a meta-team, i.e. a team where members located far away from each other.

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, [Online]. Available: http://www.cutter.com/press/050824.html

[3] T.T. Dinh-Trong, "A Systematic Approach to Testing UML Design Models. Doctoral Symposium", *in Proc. 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", *in Proc. 8th Workshop on the Semantics and Pragmatics of Dialogue, Catalog'04*, 2004, pp. 35-40.

[5] [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", in *Proc. IASTED International Conference on Software Engineering*, Innsbruck, 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.