



Introducing XP in a Large Waterfall Project

Client: CSC

Project title: LABKA II

Project domain: Application development

Business domain: Healthcare

Contrary to popular belief that complex projects involving large groups of engineers can not benefit from Agile techniques, certain Agile practices, when properly used, can make an impact even on relatively large development projects with big teams. These practices can be introduced in “mid-stream”, with little preparation, to large teams of “old school” developers who initially resist the methodology, and still bring tremendous results in productivity, product quality, and team morale in a very short period of time.

The Labka II Offshore Program

The Program team comprises over 120 staff from CSC Denmark, CSC UK, CSC France, CSC Sweden, and StarSoft Development Labs in St. Petersburg, Russia. LABKA II is CSC’s **largest Nordic Offshore Program** with over 80 StarSoft team members working in St. Petersburg. CSC has 5-10 team members also working on-site together with StarSoft. The team over the past 12 months has taken the motto of **‘One Team’** to come together to deliver the first release of LABKA II into production. In addition, the team leveraged CSC EMEA and CSC Global health solution experience from the NHS account in the UK.

StarSoft introduced XP to a large-scale software development project for Computer Sciences Corporation in Scandinavia, yielding some interesting results. A large and complex system called Labka II that had been in development for three years was about to go into production. Labka II is a medical laboratory information and production planning system supporting requisition and analysis of clinical chemical samples, blood cell analysis and the like. The system has thousands of users and communicates with a large number of advanced lab instruments. It has over 1,000 KLOC, 700+ DB tables, and 40MB+ of source code.

The Problem

However, the StarSoft team was having trouble getting performance and quality under control. Labka II is replacing the legacy Labka system, and detailed specifications were developed and handed down to the team up front. Inevitably, however, CSC and their clients (hospital systems and healthcare organizations in Scandinavia) introduced numerous changes over time to take advantage of the new technologies and include advanced functionality in the new system. The problem was that we were not very wise in how we processed the changes to the scope: we accepted the requests, without communicating back to the client their impact on the project’s complexity and timeline. At the same time, our internal project organization was not entirely efficient and was not coping with the amount of work.





Case study: Introducing XP in a Large Waterfall Project

StarSoft Development Labs, Inc.

It was becoming increasingly clear that some of the algorithms (suboptimal “patchwork” code written by separate people at different times due to poor processing of the influx of change requests over a long period) created acute performance problems, and the number of fixed defects was constantly lagging behind the number of new defects found. Both CSC and their client were unhappy because the deadline was approaching, and there just seemed to be no light at the end of the tunnel.

The StarSoft management realized that something had to be changed fast, and identified the key areas contributing to the problem:

- **Frustration from scope creep.** The individual frustration of developers was largely caused by the scope management process. The creeping scope and time pressures created the feeling among the developers that this project was never going to get “done”: no matter how hard they worked, there seemed to be little change in the overall situation.
- **Disconnect between individual effort and team results.** The numbers of defects per each developer were not so dramatic, but since this was a large project, the aggregate numbers could look depressing. People felt de-motivated because the overall project results did not seem to correlate with their individual efforts.
- **Disconnect between developers and testers.** Developers and testers didn’t talk to each other a great deal, resulting in a communication delay and lack of focus. Developers and testers were clearly out of sync, and when a defect was fixed it could take up to three or four days to get verified and fed back to the developers.
- **Fatigue from being driven too hard.** Management (all levels, from the EVP and PM to team leads) had been driving the team to work too hard, for too long, with too little result (overtime, weekends, etc). Frustration and indifference were common sentiments, and the team was unable to focus on the root causes of the problem.

New Practices

Drawing on StarSoft’s several years of experience with XP, and with the help and coaching from the group of our XP-savvy project managers, we implemented the following practices:

- **Smaller, fully functional teams.** The large team was split up into seven fully functional teams, each consisting of team lead, 3-6 developers, and 1-4 testers. Each team lead became a project manager of a small Agile project. This helped bridge the communication gap and ensured much faster turnaround.
- **Agile planning and scope management.** The project moved from “central planning” to “market economy:” developers were finally involved in determining how much will be done and by what time! Teams worked on their respective areas of the project in very short iterations (4 to 10 days), with a half-day planning game at the start of each iteration.





Case study: Introducing XP in a Large Waterfall Project

StarSoft Development Labs, Inc.

- **Individual commitment to results.** For each performance or quality target, there was now a personal commitment from a team lead. Team leads were assured by the management that scope creep would no longer be allowed, and they were effectively insulated from new change requests until the system is thoroughly cleaned up. Team leads in turn got the commitments from individual team members, on the daily basis.
- **Each day is a project.** In the morning standup meeting, tasks for the day were defined and agreed to by the team members. At the evening standup meeting, the results were also reported personally by each team member. Not only were the tasks shared, they were also written down on a dozen flipchart sheets and posted on the walls at the beginning of each iteration. (The charts were especially difficult to get some people to do. In some cases, the company's EVP had to personally oversee planning games and morning standup meetings to make sure people actually wrote tasks down on charts and posted them on walls.)
- **Granular planning and estimations on the level of hours.** Every morning, individual granular tasks were estimated (in hours) and committed to by engineers. A review followed every evening, so estimations were corrected and re-negotiated within the team almost in real time. This practice dramatically improved quality of estimations, within days of being first introduced.
- **Refactoring.** Most iterations of most teams were focused on fixing certain defects. Sometimes engineers were able to trace a certain group of bugs to a particular code fragment. Then, at the planning game, instead of deciding to simply go ahead and fix those defects one by one, the team would decide to refactor that piece of code and eliminate the root cause of the problem. As a result, we saw better code structure, better stability, and better performance (with occasional performance boosts of one or even two orders of magnitude).
- **Pair programming.** This was the practice that, while recommended by the management, was met with particular skepticism and resistance. However, eventually the infectious enthusiasm of the "XP mentors" convinced some people to try it. Immediately, it was clear that productivity did not really decrease as the "non-believers" had predicted. At the same time, the quality improved dramatically. The other benefits noted by developers were mutual education and joint code ownership. A number of experienced engineers ended up using pair programming for more complex tasks, e.g. refactoring.
- **Open communication.** XP-style communication eliminated a lot of waste and created a great atmosphere conducive to creative and productive work. The mood clearly changed from "we're working hard but it is all in vain anyway so leave us alone" to "if you know how to do things better, just tell us."

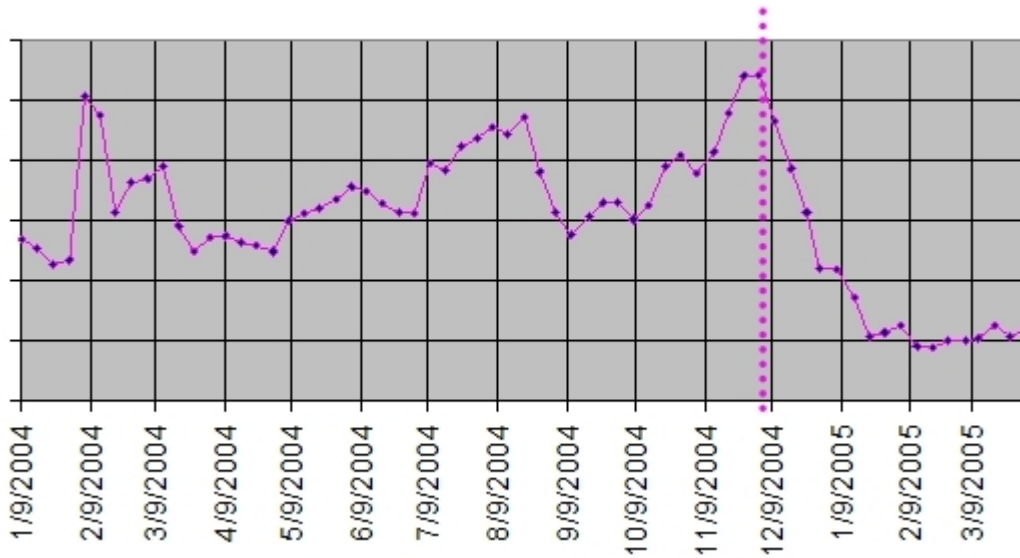




Case study: **Introducing XP in a Large Waterfall Project**

StarSoft Development Labs, Inc.

Active defects



The Results

The results were unbelievable. The metrics chart (see above) shows the dramatic decline in the amount of active defects starting in the beginning of December (precisely when the XP practices were first introduced). The quality and performance metrics charts were posted on the wall of the project room daily, and this chart has since been framed and placed on the wall in the CEO’s office as testament to the power of agility.

“The team has done an outstanding job to successfully deliver the LABKA II R1.2 product into the market. It has been a great example of Local and Global GTS-CSC and our offshore partner StarSoft Development Labs all working together to overcome numerous challenges and focusing to deliver this complex product to our first customer. The foundation of this success has allowed CSC to look to future market opportunities with the LABKA II product in EMEA and together with other CSC Healthcare products.”

Alan Guthrie
Program Manager – LABKA II
CSC

