

A) Small Online Trading Software Development

Slide 12 - SSL = Secure Socket Layer
VPN = Virtual Private Network

One cause of delay was problems with VPN and SSL. A major issue was the format of the broker reports back from the US to Switzerland after a trade was executed. It was expected that these reports would be easy to digest and pass on to individual traders to confirm that the trade was executed properly. In reality the reports came back in various formats, depending on the market on which the trade was executed. Multiple formats, fragmented execution ... grammatical analyzers had to be used, complicated, unexpected... caused delays.

Why did this project not go well?

- was there feasibility and technology evaluation done during the project preparation phase? Probably not.
- Didn't have enough expertise in house.
- (Mis)led by dominant single technical personality
- Didn't adequately write down technical requirements
- Didn't think about evolution and "staged deployment" of the system
- no usability testing, no separate testing people
- poor interfacing between business and IT worlds
- lack of separation of powers in the project
- PM too technical to understand business, VP too non-technical to interface with PM
- no schedule for hidden rework
- failed to understand that scaling up from a small prototype that can be done in a week to a "business hardened" service can be a 100 fold increase in complexity and amount of work
- security problems should be a level 1 , "live-or-die" goals consideration, essential part of the system architecture

B) Large Software Project - MS Office 2000

Advantage of new Office 2000 Organization

- individual PU's (groups) not going their own ways anymore
- shared vision

Comments on MS Stage gate process

- feature prioritization and cutting does not work if you are already on contract to an external customer (e.g. Aerospace)
- good to avoid an "all or nothing approach"
- limited for shared features, but shared features are high priority - approach works well mainly on localized features

Factors only specific to S/W projects

- importance of process, non-tangible deliverables
- software is abstract and hard to communicate to other stakeholders
- "signoff" doesn't hold because S/W is easily modified later
- different ways an organization can put pressure on a S/W project "easy to code", no robust systems engineering

Since we didn't have time to discuss the "comparison" of small vs large scale software projects in class, please find hereafter some initial personal thoughts. If you have comments and / or additions, please post them as a follow-up message in the above forum.

As the scale of a software project increases, some things become more important than others:

In small scale projects, attention must be given to the skill-set of the people involved especially in order to overcome "blocking" problems. Such technical problems that arise during software development aren't always documented. For some there exist empirical solutions while some others have just to be avoided by "working around" them.

In my experience, the best people at solving such "low-level" technical problems are those that have always taken a genuine interest in computers and in programming, the ones "who used to program their Commodore 64 when they were kids", as described in the Microsoft case video.

Therefore, in small scale projects technical expertise (superstar programmers, people who have been sitting in front of computers since the alphanumeric, green screen era) does play a very important role.

As projects become more complex, "low-level" technical problems tend to get solved

more easily, since more expensive (=better documented, more tried & tested) programming environments are used for developing, testing and debugging. However, integration issues, along with their organizational implications gain importance, since the development effort has to be fragmented into specific groups, whose work has to be synthesized at regular points in time.

Moreover, interfaces of the applications being developed with other commercial packages also take an increasingly important role.

Larger scale projects also tend to attract more attention within the development organization (since more funds are committed and therefore more people's positions are at stake) and this tends to provide more decent project monitoring and adaptations mechanisms.

However, there is a decreasing number of companies that commit to developing their own large-scale applications (in-house development), most prefer to buy / customize. The organizations that develop large-scale applications are increasingly dedicated software houses that do this as part of their core business; therefore we can assume that they capitalize on the knowledge generated from prior experiences to adapt their organizational and operational structures to efficiently manage their projects.

Significance of Staff Experience

Ion Freeman

I wanted to respond to Christos' postulate that it was the experience with language and libraries that mostly determined the success of a project, as vast amounts of time were wasted looking up API and Syntax documentation.

This may be true for small, well-defined tasks. But, facility with metaphors, patterns, approaches and platforms would seem to me to be pretty vital when decisions have to be made.

Say you're an ODBC programmer moving to JDBC. How hard is that really going to be? You're going to take a huge performance hit at first, specifying URLs, handling the driver manager, dealing with Java's particularness around exceptions and all, but you'll recover quickly. In the case of databases, the peculiarities of the dbms drive most of your coding decisions, and the syntaces are straightforward. So, learning a new database programming language shouldn't be too hard, and all the hidden complexity is in platform knowledge.