**Group members:**

**Doan Vo Quy**

**Le Hai Long**

**Tutorial 2: Need for Agile methods/framework?**

With due acknowledgements to ‘A recipe for disaster’ by: [Thomas Meloche](http://www.menloinnovations.com/founders/melochebio.htm), The Menlo Institute LLC, [www.menloinstitute.com](http://www.menloinstitute.com)

**Yet another development project failure**

Recently I found myself at dinner with a group of six or seven software developers who were working on a project for a large retail company. Although the food was good, the conversation was depressing.

These developers, all excellent in their field, were working on a large project designed to “integrate multiple information systems, both locally and globally, within an intranet-based digital framework”.

The project is highly buzz word compliant: distributed systems, object-oriented, Internet, Intranet, and incremental development are all likely to be heard. Buzzwords don't mean anything.

**Recipe for Disaster**

The developers described a project that was a recipe for disaster:

* The team size was over 150 people.
* The development groups were isolated from one another - one group was doing analysis, one group was doing software design and a separate group was doing implementation.
* They were using incremental development, but not iterative development.
* No feedback existed from one increment to the next increment.
* No feedback existed from development to requirements, analysis or design.
* Users were very rarely consulted on the development of the system.
* Testing appeared to be non-existent – developers tested their own code but not part of an integrated testing strategy.

The list of problems I heard in just a few short minutes could go on and on. All of the developers were trying to do a good job but the process made it difficult at best, and impossible at worst. The developers were going crazy and all wanted out - in fact the dinner meeting was about helping them find new jobs.

**Death March**

The developers could see that the current project development and management practices were a recipe for disaster. Why couldn't the managers? The developers had been working with software long enough to know that the entire effort would probably fail and be cancelled. They weren't interested in working any longer on a death march.

No one likes developing software that is never delivered, that is never used. The sad thing is the large number of software developers who have worked in the industry for years and years, and have never seen even one of their projects actually ship to a customer.

These developers were desperately looking for another project; a project which could succeed. A project where they knew they could do good work and deliver real value. Top developers are always looking for a place where they can actually succeed. This is the reason why even in the midst of the tightest job markets we have never had any trouble hiring them.

Can a project with a multi-million dollar budget and over 150 staff fail? You bet. Ironically, the statistics demonstrate that the larger the budget and the bigger the team, the more likely the project is to fail1. Nine out of ten large projects experience severe problems. With regard to software development projects, bigger is almost never better. Yet companies still insist on developing large software projects with large teams that they clearly do not know how to manage.

1Standish Group Chaos Report [2001]

**Tutorial questions**

1. ***The author of the above article notes that “the development groups were isolated from one another - one group was doing analysis, one group was doing software design and a separate group was doing implementation”.***

***Why would this arrangement be a recipe for disaster?***

This setup is a disaster waiting to happen because it follows a "waterfall" approach with almost no communication or feedback between stages. For a project to succeed, teams must work together and stay in sync. In fact, even for a project to be acceptable or barely passable, the teams involved in development need to collaborate. It’s obvious—the output of one group is the input for the next. How can the design team create a good design without understanding the analysis results? How can the implementation team know what to do if they don’t coordinate with the other two? Clearly, no project has ever succeeded under such conditions.

Some might argue that the analysis phase comes first and produces results for others to use, so it can be done separately. But that’s not true. Requirements often change throughout development, sometimes multiple times. When that happens, re-analysis is unavoidable, making the idea of isolating the analysis phase pointless. And if requirements change rapidly, the analysis team can even use past results from other teams to support their work.

***2. The author also quotes the Standish Report stating “the statistics demonstrate that the larger the budget and the bigger the team, the more likely the project is to fail1. Nine out of ten large projects experience severe problems. With regard to software development projects, bigger is almost never better”.***

***Why do you think that large projects with big budgets are more likely to fail?***

Large projects with big budgets often fail because many people don’t have enough time for proper training, which leads to mistakes. Too many people also mean too many ideas, and that can cause disagreements and slow down progress. Managing a large team is also difficult because it’s hard to keep everyone on the same page.

A big budget can also lead to failure because it often creates a false sense of security, making teams spend money on unnecessary things instead of focusing on real priorities. When there is too much funding, teams may become less careful with planning and resource management, leading to inefficiency and waste.

3. ***The author mentions both incremental development and iterative development within the case study.***

1. ***Explain both of these terms***
2. ***Why would the approach taken to incremental and iterative development described in the case study contribute to the project failure?***

a. Incremental development and iterative development are both methods used in Agile. Incremental development means building the project step by step, with each part being completed before moving on to the next. On the other hand, iterative development focuses on creating an initial version of the product and continuously improving it through multiple cycles based on feedback.

b. The project in the case study only used incremental development but did not apply iterative development, which caused many issues. Without feedback from one phase to another, teams worked in isolation and had no opportunity to adjust or improve based on real-world results. Specifically, since there was no feedback between development stages—from analysis to design to implementation—early mistakes were not corrected in time, leading to bigger problems later on. If both incremental development and iterative development were used together without proper management, they could conflict with each other, making the development process chaotic and ultimately leading to project failure.

***4. The case study tells us that: “Testing appeared to be non-existent – developers tested their own code but not part of an integrated testing strategy”.***

***Why is this flagged as a problem? What would be a more effective way of testing within a development environment?***

A project without proper testing cannot be successful. Developers may test their own code and say there are no bugs, but that only proves the code works within the small part they are responsible for. There is no guarantee that issues won’t appear when different modules are combined. The only way to ensure this is by having an integrated testing strategy.

Additionally, no matter how careful a developer is, they cannot predict every possible situation that might cause a bug. This is why dedicated testers are needed. With their expertise, they create structured test plans to check for errors that developers might overlook. A proper testing process helps catch problems early, making the project more stable and reliable.