**Systems Development Methods & Frameworks (COMP1787)**

**Tutorial: Why do we need methods?**

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. How depressing.

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]