



The Menlo Institute LLC

Not Enough Time for Training

A recipe for disaster

**Thomas Meloche © 2002
Fellow, The Menlo Institute**

**The Menlo Institute LLC
212 N. Fourth Ave
Ann Arbor, MI 48104
www.menloinstitute.com
(734) 665-1847**

Yet Another Failure

I found myself at dinner with a group of six or seven software developers who were working on a project for a large manufacturing company. And, although the ribs were good, the conversation was depressing.

These developers, all good engineers, were working on a large project designed to integrate multiple information systems both locally and globally.

The project is highly buzz word compliant: distributed systems, COM, 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 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.
- There were no test frameworks.

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 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 where they could succeed. A project where they knew they could do good work and deliver real value. Top engineers 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.

Not Enough Time For Training

Almost weekly we run into a new story like this one. Typically the participant needs to describe just one or two features of their process to make it clear to us that the project will not succeed.

We personally know teams that have spent tens of millions of dollars on a software project before finally giving up and abandoning the project. This is inexcusable given what we now understand about how to successfully develop software.

Here is a suggestion: before starting your next development initiative spend a few days and a few hundred dollars studying why projects succeed or fail. The Menlo Institute offers a one day course covering the Secrets of Software Success. At the end of this class you will be equipped to assess if your project is more likely to succeed or fail. You will also learn how to save the project if you determine it is likely to fail.

Of course, the excuse we often hear about why people don't study software success practices is that "we don't have enough time for training." What do you think? Do you have enough time for training?

¹ Standish Group Chaos Report [2001]

About the Menlo Institute

Founded in 2001 by Thomas Meloche, Richard Sheridan, James Goebel and Robert Simms, The Menlo Institute specializes in Software Development Process and Methodology with a special focus on an agile software development environment called The Software Factory.

The Software Factory

The Software Factory is a complete agile software development methodology created at Menlo. The Software Factory combines best practices from Alan Cooper's Interaction Design, Kent Beck's Extreme Programming, The Rational Unified Process and Six Sigma. The Menlo Institute has unique insight from real-world experience on how these practices can work together effectively.



**Thomas Meloche
President
The Menlo Institute**

Thomas Meloche is President of The Menlo Institute LLC and a founding member of Menlo Associates LLC. His passion is software development best practices and building successful software teams. Before founding The Menlo Institute, Thomas was the Director of Engineering at AppNet, Inc. For over seven years, he led a consulting staff that grew from

30 people to over 240 people which generated revenue of over 100 million dollars. As a result, he is intimately familiar with the real problems that befall real software development projects. He also knows how to rectify them. During his tenure his delivery organization had a remarkable 100% successful delivery record.

Thomas has a B.C.E. in Computer Engineering from the University of Michigan, and especially enjoys providing training back to his alma mater. He has taught classes and provided mentoring for both the University of Michigan Law and Medical Schools.

Menlo Institute Classes by Category		
Extreme Programming (XP)	Foundation Courses	Rational Unified Process (RUP)
Introduction to XP Coaching XP Teams Writing Story Cards XP BOOT Camp / Immersion	Secrets of Software Success Six Sigma Software Building a Software Factory Object Technology Overview	Managing with RUP-Lite Capturing Reqs With Use Cases Practical UML RUP BOOT Camp / Immersion
Requirements Gathering	Object Technology	Project Management
Capturing Reqs With Use Cases Writing Story Cards	Object Technology Overview Obj. Oriented Analysis & Design Advanced Obj. Oriented Design	Managing with RUP-Lite Software Project Mgmt Six Sigma Software