

How to fail a project

Thomas Sundberg

Developer for more than 20 years

Masters degree in Computer Science from the Royal Institute of Technology, KTH, in Stockholm, Sweden

I write computer programs

@thomassundberg

tsu@kth.se

<http://thomassundberg.wordpress.com>

Goal

Describe a set of patterns that will assist you to fail a project



Organisation

Division

Analysis

Architecture

Development

Configuration management

Test

Operation

Line and projects

Line management

Divide employees between projects

Company policies

Enforce the usage of common framework

One tool shall solve all problems

Force the usage of the same development environment

Working offsite

Make it extremely difficult to work from home

No laptops

No VPN

No external ssh

Control

All details

Never self organise

Measuring

How many bugs have one person fixed

Test coverage

Improvement

No regular retrospectives

Management should define all improvements

Personal development

Hire personnel with the proper skills

Don't pay for any courses

Never send people on conferences

Don't sponsor any books

Learn new things on personal time

Demotivation

Criticize in public

Don't invest in tools

No large screens

Managers should have the fastest computers

Resources

Refer to people as resources

Resources can be replaced

Demand overtime

40 hours a week is for wimps

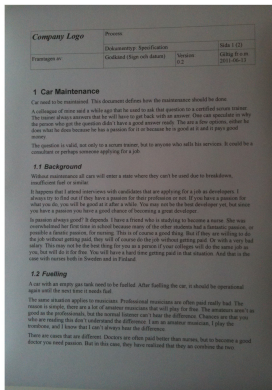
Avoid a clear goal for the extra effort

Hero culture

One person who solves all problem

Communication

Bandwidth



Written

Prefer

Can be read at any time

Meetings

Schedule lots of meetings

Prefer long meetings

Demand that everyone attend

Cancel them late

Don't show up

Show up unprepared

Long meetings

Avoid decisions

Schedule another meeting for decision

Daily stand up

Standup meetings are just uncomfortable

Weekly dev meetings

Long

Cancel them often

MoM

Always demand written MoM after every meeting

Mail

Always top post

Always high priority

Add all recipients in the to field

Add more people to a thread

Wrong subject

Specifications

Always written

Avoid examples

A specification

<i>Company Logo</i>	Process:		
	Dokumenttyp: Specification		Sida 1 (2)
Framtagen av:	Godkänd (Sign och datum)	Version: 0.2	Giltig fr o m: 2011-06-13

1 Car Maintenance

Car need to be maintained. This document defines how the maintenance should be done.

A colleague of mine said a while ago that he used to ask that question to a certified scrum trainer. The trainer always answers that he will have to get back with an answer. One can speculate in why the person who got the question didn't have a good answer ready. There are a few options, either he does what he does because he has a passion for it or because he is good at it and it pays good money.

The question is valid, not only to a scrum trainer, but to anyone who sells his services. It could be a consultant or perhaps someone applying for a job.

1.1 Background

Without maintenance all cars will enter a state where they can't be used due to breakdown, insufficient fuel or similar.

It happens that I attend interviews with candidates that are applying for a job as developers. I always try to find out if they have a passion for their profession or not. If you have a passion for what you do, you will be good at it after a while. You may not be the best developer yet, but since you have a passion you have a good chance of becoming a great developer.

Is passion always good? It depends. I have a friend who is studying to become a nurse. She was overwhelmed her first time in school because many of the other students had a fantastic passion, or possible a fanatic passion, for nursing. This is of course a good thing. But if they are willing to do the job without getting paid, they will of course do the job without getting paid. Or with a very bad salary. This may not be the best thing for you as a person if your colleges will do the same job as you, but will do it for free. You will have a hard time getting paid in that situation. And that is the case with nurses both in Sweden and in Finland.

1.2 Fuelling

A car with an empty gas tank need to be fuelled. After fuelling the car, it should be operational again until the next time it needs fuel.

The same situation applies to musicians. Professional musicians are often paid really bad. The reason is simple, there are a lot of amateur musicians that will play for free. The amateurs aren't as good as the professionals, but the normal listener can't hear the difference. Chances are that you who are reading this don't understand the difference. I am an amateur musician, I play the trombone, and I know that I can't always hear the difference.

There are cases that are different. Doctors are often paid better than nurses, but to become a good doctor you need passion. But in this case, they have realized that they can combine the two.

<i>Company Logo</i>	Process:		
	Dokumenttyp: Specification		Sida 2 (2)
Framtagen av:	Godkänd (Sign och datum)	Version: 0.2	Giltig fr o m: 2011-06-13

1.2.1 Deviations

Not all cars use petrol, some use diesel. It is important that this difference is honoured.

So what is your passion? Try to find an answer to that question and describe it when you are selling yourself. That may be as a consultant, when you apply for a job or anything else. But make sure that you get reasonable paid for your passion, otherwise you destroy for yourself and everyone else with a similar passion.

What is my passion then? At this point in my life it is development and teaching about development. I want to develop things in a good way. That translates into fast feedback loops and experimenting my way forward. This in turn translates into using eXtreme Programming as much as possible.

My passion may change. But this is something I expect. A static life isn't something I would want to live. I have realized that I'm always headed somewhere professionally and that suits me just fine.

So, ask yourself what your passion is next time you try to sell your services and make sure that it is your passion that you sell.

1.3 Test cases

To be defined.

A specification

1.3 Test cases

To be defined.

Common language

No need for a common language

No need for system metaphors

The same view of the system

Customer

Avoid having the customer at the same premisses as the developers

Definition of done

Everybody knows when a functionality is done

Separate dev and ops

Devs should not be allowed into the production systems

Devs should develop deployment packages and scripts

Devs should be responsible for bug fixing

Planning

Planning

Plan 12 months ahead with all details up-front

Follow the plan strict

Collective planning

Never!

One person should estimate all tasks

Planning poker is just a waste of time

Estimation

Force estimates onto people

This really can't take more than 4 hours?

Design

Committee

All important design decisions

No regular meetings

Unusual frameworks

Home made

Commercial

Ivory tower

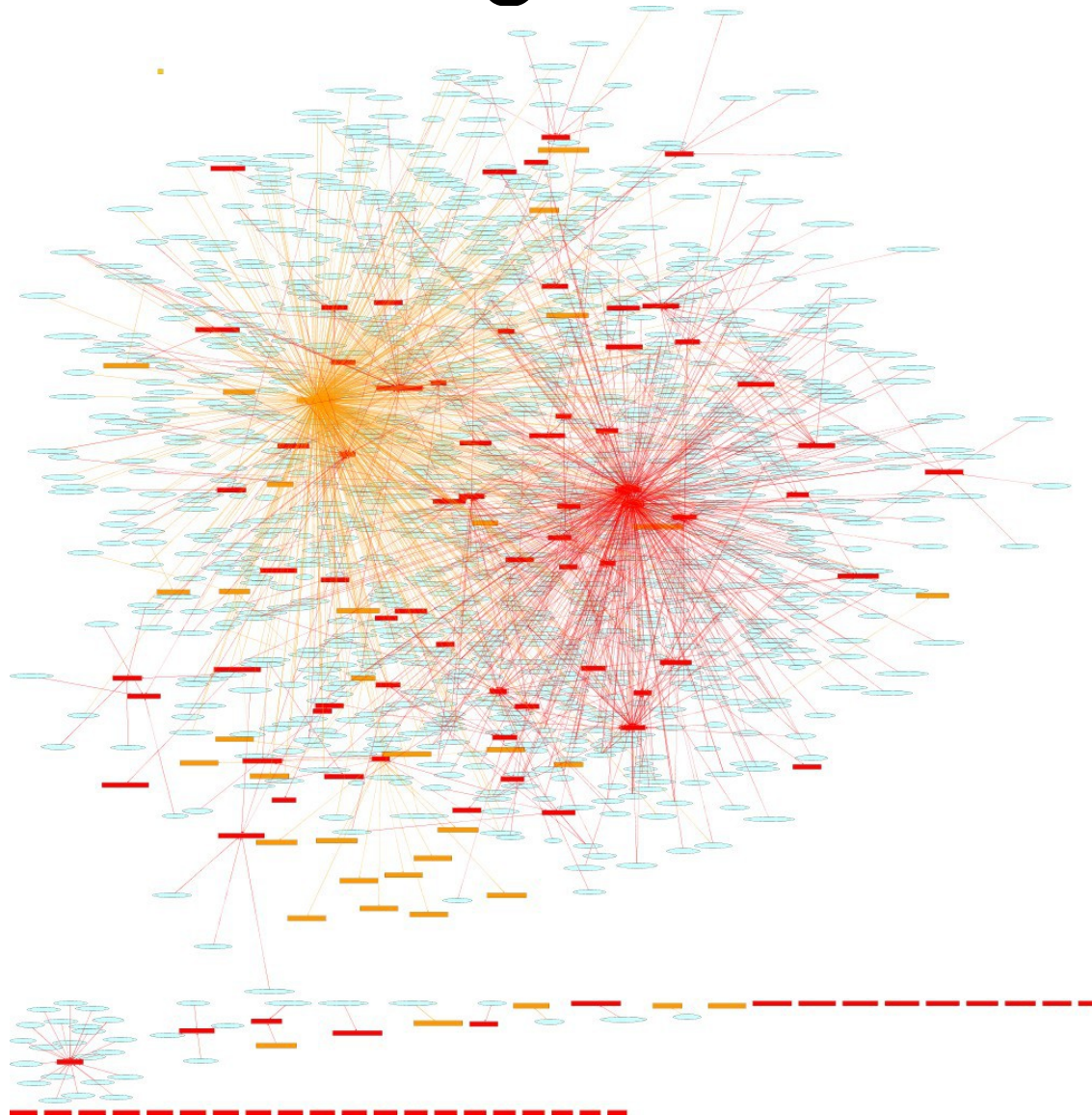
Only employ ivory tower architects



Singletons

Supports global variables and state

Singletons



<http://blog.code-cop.org/2012/01/why-singletons-are-evil.html>

Non functional requirements

Define load test as

“Must cope with high load”

Security

Add last

System communication

Must be system agnostic

Prefer xml over http

Simple design

Is for simple systems

Complex systems demand a complex design

Four rules of simple design

No need to follow

All test passes

Reveal intent

No duplication

Small

Colours

Do not be afraid to use colours

Red and green are great colours to mix

Development

Coding standards

Should be defined by a management committee

Version Control System

A shared disk should be enough

Clear Case

Code ownership

One developer owns the code in each module

Forbid other developers to edit that code

Refactoring

Avoid re-doing things

Get it right the first time

Refactoring

Complicate the code whenever possible

Continuous integration

A waste of resources

Have developers chase down missing files

Configuration Manager

One CM should be enough

No automation

Build manual

Priorities

Largest project always most important

The project with most people

Smaller projects have to adopt

Wait

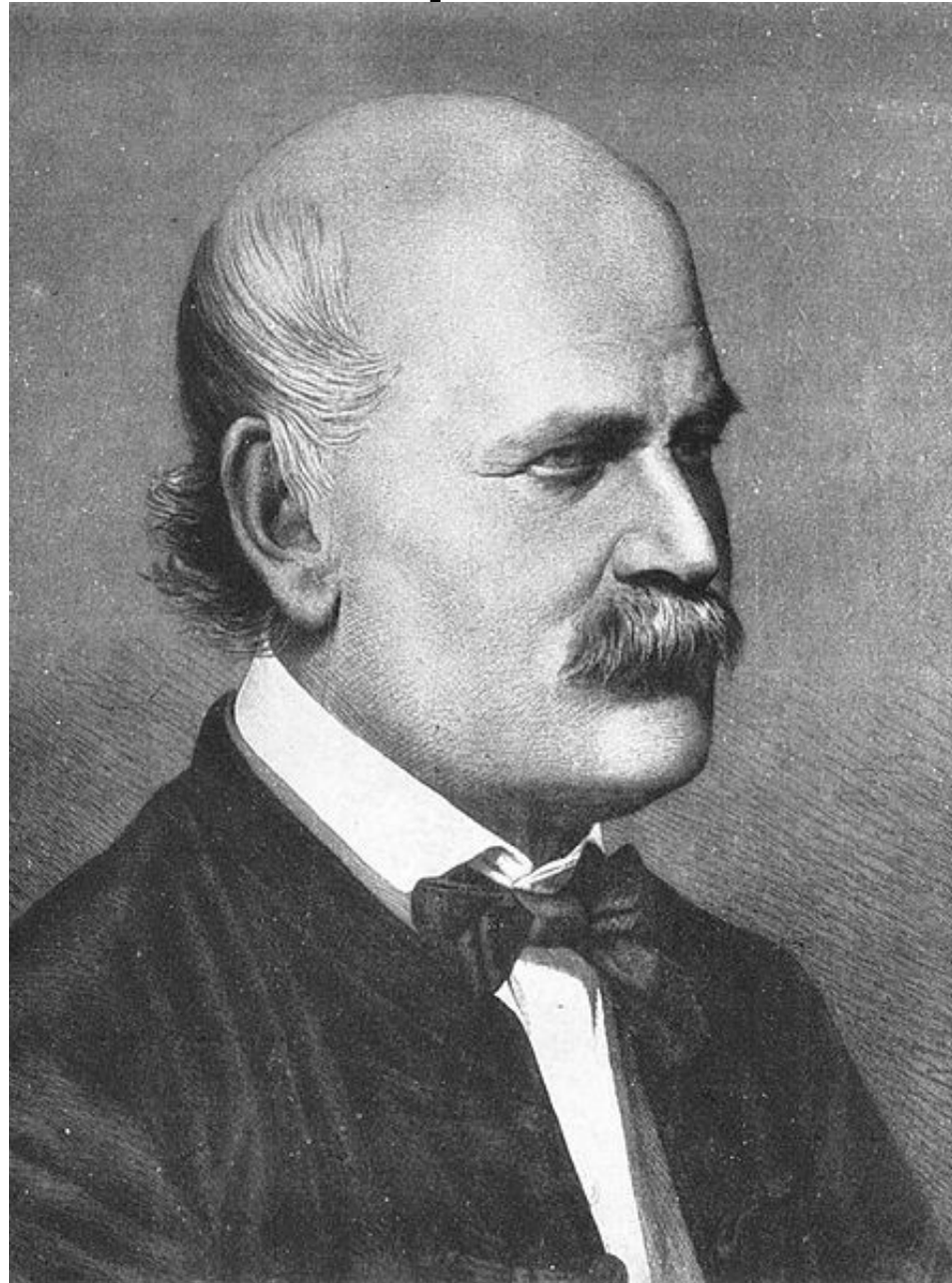
Rush

Accept broken builds

Systems sometimes doesn't compile

Do not stop the line to fix the build

Good practices



Pair programming

Inefficient

Expensive

Forbid TDD

Testing should be added last

100% test coverage

Anything less is a failure

Javadoc

Demand 100% javadoc

Testing

Add quality at the end

Quality should be assured last

Testing early will test things that is not done yet

Performance can be added at the end

Functional testing

Manual

Through the user interface

Corner cases

Focus on corner cases

The most common cases will be tested anyway

Easy parts first

Postpone testing of difficult cases as long as possible

Stable areas

Prefer to test stable areas to avoid hassle with bug reports

Load test

Should be done when the system is ready

Use the right tools

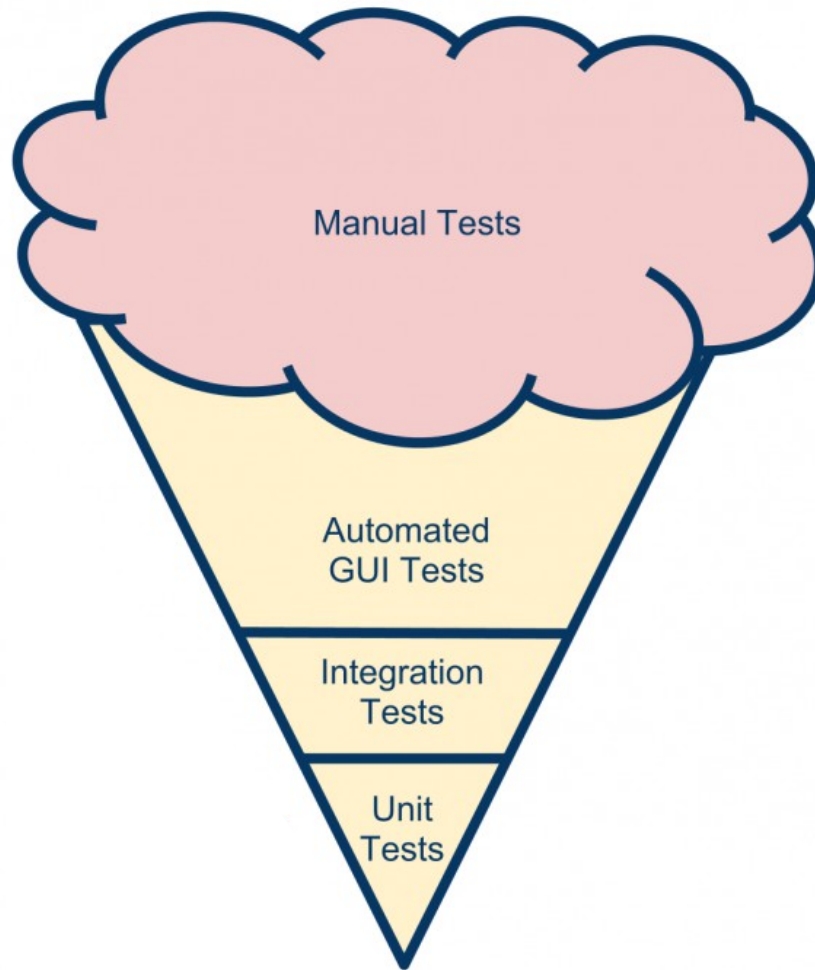
Automation

Testing should be done manually

It is too important to be left to a computer

Only a human being can judge a user interface

Automation



Production

Avoid deployment to production

Risky and error prone

Different environments by different people

Reduce the number of times

Maximum 2 times a year

Never any small releases

Continuous deployment

Deployment is risky

Has to be done manual



Not

All I have said here is wrong

Negate everything

Agile manifesto

Individuals and interactions

over processes and tools

Working software

over comprehensive documentation

Customer collaboration

over contract negotiation

Responding to change

over following a plan

Principles behind the Agile manifesto

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity - the art of maximizing the amount of work not done - is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

XP principles

Fine scale feedback

- Pair programming

- Planning game

- Test-driven development

- Whole team

Continuous process

- Continuous integration

- Refactoring or design improvement

- Small releases

Shared understanding

- Coding standards

- Collective code ownership

- Simple design

- System metaphor

Programmer welfare

- Sustainable pace

Resources

Agile manifesto - <http://agilemanifesto.org/>

XP - <http://www.extremeprogramming.org/>

My blog - <http://thomassundberg.wordpress.com/>

How to fail a project

Thomas Sundberg

Developer for more than 20 years

Masters degree in Computer Science from the Royal Institute of Technology, KTH, in Stockholm, Sweden

I write computer programs

@thomassundberg

thomas.sundberg@waymark.se

<http://thomassundberg.wordpress.com>