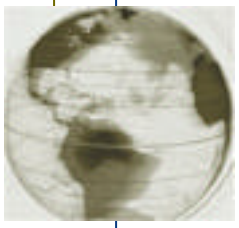


Alternative approaches
Agile processes

Sesam

Stockholm, 021023

Even-André Karlsson

Doc. number LD/QLS 00:0001
Revision PA1
Date 1 September 2000
Approved NN
Q Template 6.0

Q-Labs

Agenda

- Q-Labs and personal background
- What is special about software?
- State of the practice
- People, Process, Tools, Technology
- Agile processes - eXtreme programming

The Q-Labs Company

- Q-Labs Manages Software Risks with World Class Customers
- Global Reach

  Offices in:
  - France
  - Denmark
  - Germany
  - Ireland
  - Norway
  - Sweden
  - UK
  - USA

- 155 Employees
- Highly Qualified Staff - many with long term industrial background
- A Broad International Client Base, e.g.
  - Alcatel, Axis, Ericsson, France Telecom, Telia, Telenor
  - ABB, R. Bosch , EDF, IBM, Siemens, Schneider Electric, Thomson Detexis, Volvo
  - Atomic Energy Board of Canada, FAA, Norwegian Ministry of Justice, Swedish Civil Aviation Administration
  - FMV, US Army TACOM

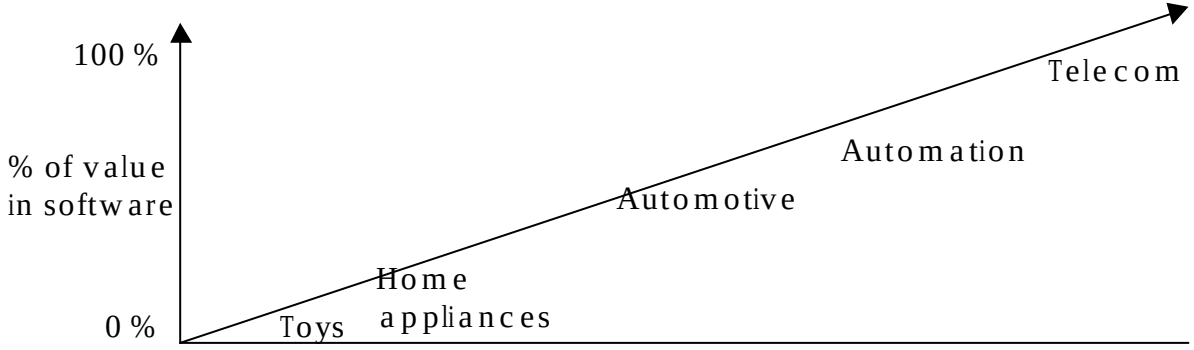
  Provides state-of-the-art and industrial-best-practice solutions with quantifiable results
- Owned by Ratos, Ericsson and Det Norske Veritas (DNV)

Professional background

- Ph.D. in computer science, Norway, 1990
- Researcher, US & Esprit, 1990-1992
- Principal consultant Q-Labs
  - Ericsson, ABB, Bosch, Alcatel,Lucent, Philips
  - Esprit project REBOOT
  - Many aspects of software engineering
    - Development tools
    - Team work
    - Architectures
    - Incremental development
    - System strategy

What is special about software?

- ”Lack” of ”production” - ”only” design
- Rate of change, Internet year, Murphy’s law
- Complexity is increasing
- Extent of software in products



State of the practice: Immature

- Financial risks
  - Cost overruns
  - Not meeting requirements
  - Cancellations
  - Delays
- Failures

USS Yorktown, 1997

Ariane 5, 1996

AT&T Frame Relay Network Outage

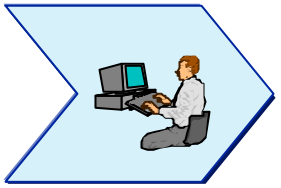
What to address?

Tools

Technology

People

Process



Tools

- Now in fashion again (last time in 80's)
- Two main consolidated suppliers:
  - Rational: RequisitePro, Rose, ClearCase
  - Telelogic: Doors, Tau, Continuus
  - Runner ups: Microsoft, Oracle
- Problems/issues
  - Do they scale up
  - Maintenance and enhancement
  - Integration between phases and tools
  - New standard for UML real-time

Process

- Three approaches:
  - Reference models: CMM, ISO, TickIt, Spice
  - Process solutions: Rational Unified Process
  - Minimal processes: eXtreme Programming
- Problems/issues
  - Large efforts involved in process tailoring
  - Fitting process to problem
  - Generally moving to incremental development
- Steady focus, but shifting emphasis

People

- Boehm identified this as the main impact
- ”Recent” advances
  - People Capability Maturity Model
  - Personal Software Process
  - Team Software Process
  - DeMarco and Lister: PeopleWare
- Generally overlooked

Agile processes

- People focus: Developer
- Process focus: Support programming
- Tools: Code is king

Modeling doesn't make me nervous!

It's the model!

The role of models

Should it really look like this?

Requirements ReqPro/Doors → Business model, UML → Use case model → Analysis Model → Design model → Code

Why not like this?

Code, test, → Code, test, → Code, test

Model/Document

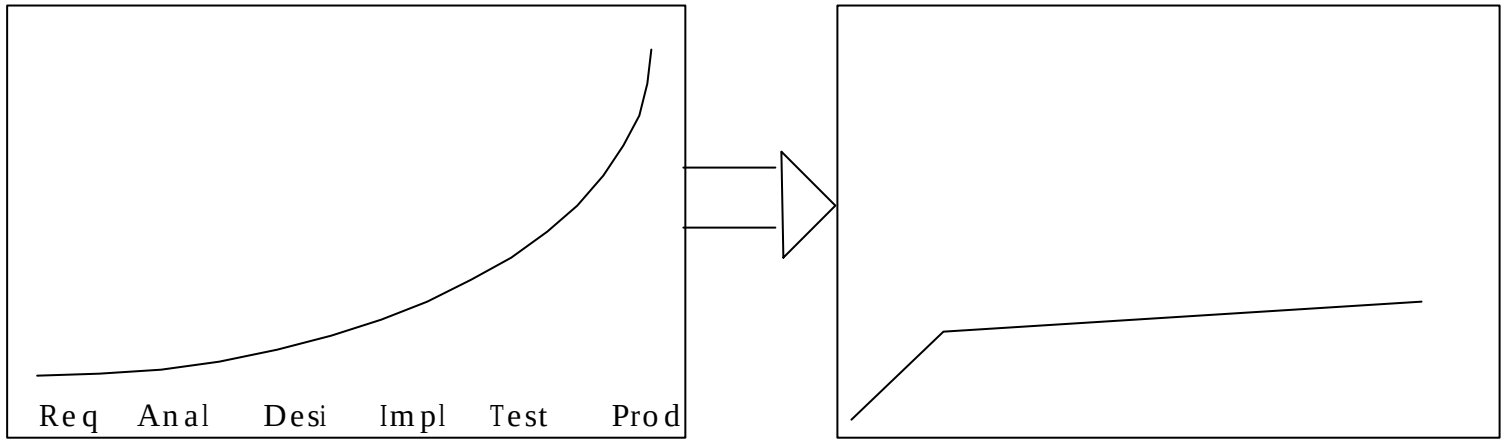
XP

- Motivation
- Process
- 12 Principles

Takes good things to extreme

- If reviewing is good we will review all the time => Pair programming
- If testing is good, everybody will test all the time => automatic testing, writing tests as specfications, parallel customer test
- If architecture and design is good, we'll make it part of everybody's daily business (metaphor and refactoring)
- If simplicity is good, we'll always make the simplest thing that can run all test cases
- If integration testing is good, we will do it several times a day
- If short iterations are good, we will do them very short - minutes and hours not weeks and months

The cost of change



- A simple design with no extras
- Automated test will reduce lengthy studies - try and see
- Lots of practice in modification (refactoring)
- Only one level of documentation - the code

Back to basics

- Coding - otherwise there is nothing
- Testing - tells you when you have finished coding
- Listening - to know what to code and test
- Designing - to keep coding, testing and listening for ever

XP process

- Planning
  - User stories
  - Planning game
  - Iterative development
- Design
- Coding
- Testing

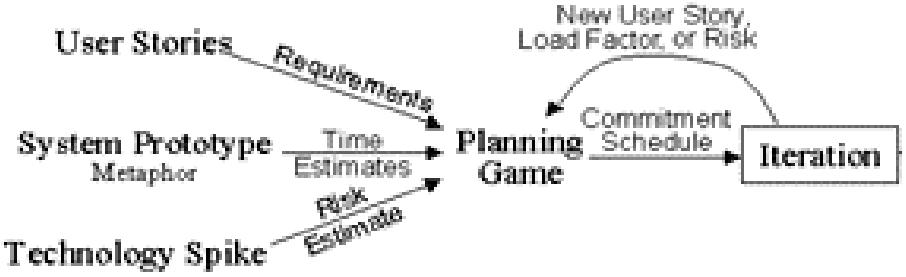
Planning: User stories

- Customer writes what system must do for them
- Not detailed, 3 sentences
- Basis for
  - Effort and risk estimates
  - Function test cases

- Every user story shall take 1-3 weeks to implement

Planning: Planning game

- Business and technical people participate
- Estimate each user story in terms of effort, risk and priority
- Make a card of each user story
- Plan small and frequent releases



Planning: Iterative development

- Start every iteration with a meeting, and plan what shall be done
- Break user stories into design activities
- Forbidden to implement something that is planned for a later iteration

Design

- Simplicity is rule number 1
  - Always the simplest possible solution
  - If something can be simplified - change it.
- Design is done together - CRC cards
- Make spike for difficult technical problems
- Never add functionality you believe will be needed

Coding - 1

- Customer always there
  - Participate in team
  - Detail user stories when needed
  - Function test and approval of system
  - A lot of time, but efficient in long run
- Write test case before code
  - Forces developers to detail requirements
  - Forces writing of testable code
- Pair programming: Efficient and quality

Coding - 2

- Sequential releases: Focus on one iteration
- Integrate often, several times a day
- Collective ownership of code: all can change code
- Optimise at the end: First Work, so Right then Fast

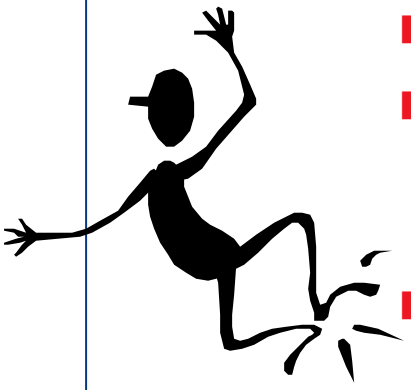
Testing

- Unit test
  - Automatic test suites
  - Ensures that ”everything” is tested
- Function test
  - Based on user stories
  - Automatic test suites

12 principles

- The planning game
- Small releases
- Metaphor
- Simple design
- Testing
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration
- 40 hour week
- On site customer
- Coding standards

Why should we work this way?



- Easy to see progress - good project control, fast feedback and high motivation.
- Always a stable system (good quality assurance).
- Easier to find faults and less re-work.
  - New faults are introduced during the last 24 hours.
  - Designers and testers work together - direct feedback.
- Less documentation - avoids several levels of hand over, interpretation and re-writing.
- System competence - increased flexibility and commitment.

Agile processes and modeling

- Compatible if one model replaces code
  - SDL/SDT
  - RoseRT
  - UML 2.0 - Realtime
- Modeling is OK on the whiteboard
- Complete models apart from code are not agile
  - Requirements models (req database, elaborate use cases)
  - Analysis and design models (standard UML)