A global software project for a globalized world

Bertrand Meyer

ETH Zurich and Eiffel Software

CSEE&T, Charleston, April 2008

ETH
Chair of Software Engineering

Today's software development

Gone are the days of one-company, one-team, one-location projects

Today's ecosystems are multipolar

- Distributed team
- Flexible assignment of tasks
- Outsourcing, insourcing, backsourcing
- Flexibility is key: the world belongs to the nimble
- Lots of ideas, proven and unproven, e.g. agile methods
- What happens in the absence of direct contact?

Little of this is taught in universities!

2

ETH Zurich

Only federal university in Switzerland
(with sister institution, EPFL)
Created 1855
Associated with close to 30 Nobel prizes

Einstein, Pauli, Clausius, Bernays, …

In computer science: Niklaus Wirth; birthplace of Pascal, Modula-2, Oberon, Lilith

About me: at ETH since end 2001, from industry

3

Software engineering courses at ETH

1. ***Software engineering***

   3rd year course, quasi-required

2. Software Engineering for Outsourced and Offshore Development

Elective course (bachelor's/master's)

Since 2004, with Peter Kolb
First course of its kind anywhere (I think)
Follows from *IEEE Computer* paper, January 2006
Used to serve as general soft. eng. course

4

Specific context

"Bologna" reform in Europe: bachelor, master, PhD

At ETH: influx of outside students in year 4 (master's)

5

Recent evolution of outsourcing course

From

- "Software Engineering for Outsourced and Offshore Development"

To

- "Distributed and Outsource Software Engineering (DOSE)"

Turning into

(Distributed) (software engineering course)

as well as

(Distributed software engineering) (course)

6

Digression: industry practice

Eiffel Software

Company based in Santa Barbara, California

- Used to be president, since 2001 Chief Architect
- More active role in project management since 2006

Focused on O-O tools, Eiffel approach, Design by Contract

EiffelStudio development

Two-million line code base (almost all Eiffel, a bit of C)
Major industry customers, mission-critical applications
International standard: ECMA and ISO
Open-source license, same code, vigilant user community
6-month release schedule since 2006

Developer group:

- Small group (core is about 10 people)
- Most young (25-35)
- Highly skilled
- Know Eiffel,O-O, Design by Contract
- Strong company culture, shared values
- Know environment, can work on many aspects
- Distributed
- Mostly, we live in a glass house

9

Distributed development principle 1

I would not try unless people have previously worked together in a common location

10

Distributed development principle 2

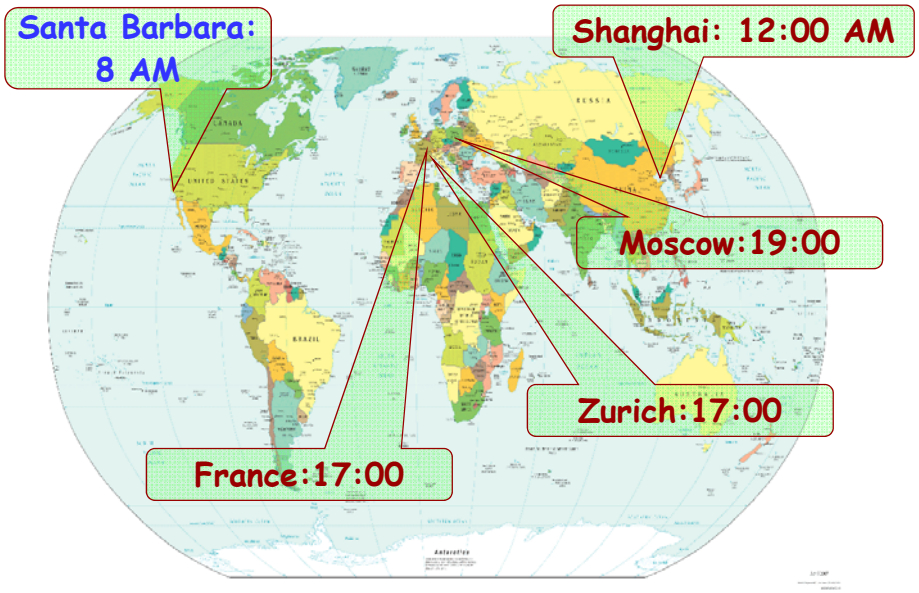
Email is great, but every team needs contact

Our solution: the weekly one-hour meeting

Replaced a SB-only meeting (every Friday, until 2005)

11

How do we organize a meeting?



12

Meeting properties

Top goal: ensure that we meet the release deadline

Tasks: check progress, identify problem, discuss questions of general interest

Not a substitute for other forms of communication

Time is strictly limited: one hour come rain or shine

13

Meeting tools

Webex for conference call management

X-Lite as a replacement for Skype

Google Docs

Wiki site (http://dev.eiffel.com)

Skype: chat window only

14

Communication

Works, but still not perfect

Still too much non-semantic communication

Audio communication heightens problems, e.g. accents

Need to work after the meeting

Ability to edit a common document in real time is a critical advantage

Documents are key: mix of verbal and written word

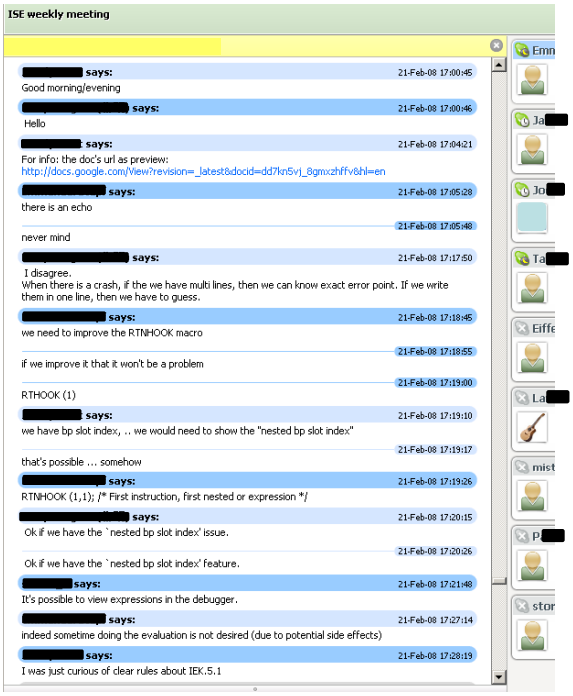
15

Distributed development principle 3

Infrastructure matters

Connection problems are not fun after the second time

16



17

Rule 4

Scripta manent

(Or: talk is cheap)

18

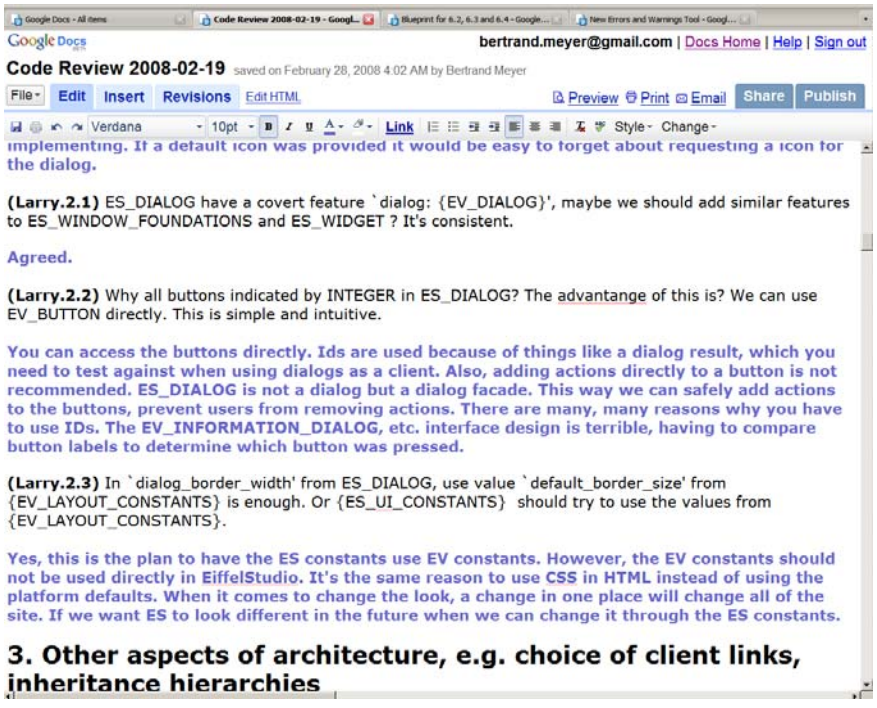
Recent addition: the remote code review

Traditional: time-consuming, tedious, questionable value

With the Web it becomes much more interesting!

- Classes circulated three weeks in advance
- Comment categories:
  - Choice of abstractions
  - API design
  - Architecture choices
  - Algorithms & data structures
  - Implementation
  - Programming style, comments, documentation
- Comments in writing on Google Doc page, starting one week ahead
- Author of code responds on same page
- Meeting devoted to unresolved issues

19



20

End of industry digression

Software engineering courses at ETH

1. ***Software engineering***

   3rd year course, quasi-required

2. ***Distributed and Outsourced Software Engineering (DOSE)***

   Elective course (bachelor's/master's)

Principles of teaching software engineering (1)

Principles for the course as a whole:

1. Focus on the non-programming aspects

"DIAMO":

- **D**escribe: specify (systems, designs, implementations…) and document
- **I**mplement: build the products; this includes design as well as programming
- **A**ssess: verify, validate, analyze, test, measure (products & processes)
- **M**anage: organize work, communicate, collaborate
- **O**perate: deploy systems & oversee proper functioning

23

Principles of teaching software engineering (2)

Principles for the project:

1. Set up a controlled environment to exercise the aspects you want to emphasize
2. The project should include programming
3. Aim for a deployable project
4. In a group project, divide work by subsystems (cluster), not lifecycle task
5. You design and implement what you specify
6. Learn to design a test plan for someone else's software

24

Goal of the DOSE course

Prepare students to the new, globalized world of software development

Some topics:

- Requirements in a distributed project
- Quality assurance
- Project models, CMMI
- Agile methods
- Managing relationships with suppliers, contract negotiation
- …

25

Course characteristics at ETH

Elective course, bachelor's and master's

Typically 15 to 35 students

Many with industry experience

2 lecture hours (Wednesday 8 to 10), one exercise hour mostly devoted to the project

26

Project: involving other universities

Since 2007:

- Odessa National Polytechnic (Ukraine)
- University of Nizhny Novgorod (Russia)
- University of Zurich

27

Why we are doing this

Distributed Software Engineering raises new challenges

Techniques exist, but the skills must be taught

Students initially have no idea of the issues, but they understand them quickly and (painfully) find solutions

This is also a great way to teach by example the benefits of software engineering principles, e.g. abstraction, API design, documentation, requirements…

28

Project principles and roles

Emulate industrial setting, but only where it makes sense

- Benefits of a controlled setting
- Goal #1 is to learn

All groups created equal

- We do not want one university to specify & another implement

Clear management structure

- Central management role, currently at ETH
- Technology choices imposed
  - Eiffel (as a language and method)
  - Origo software development platform
    - origo.ethz.ch
  - Web tools
  - Any others that may be necessary
- Universities can contribute, e.g. broadcast own lectures

29

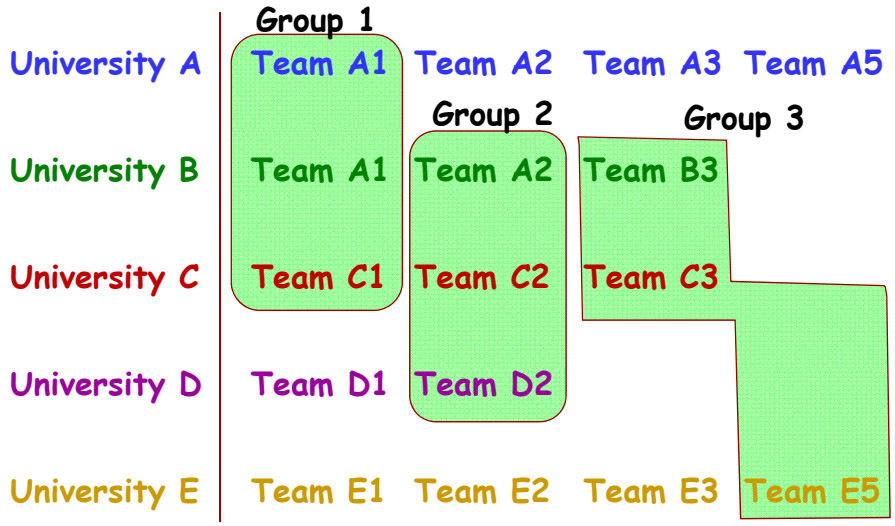
Organization

Cluster-based, not process-based

- A team includes a few students (2 or 3) from one university
- A group is a collection of groups, each from a different university
- Each group does full project
- Each team does a part of the project
- This is a part of the system ("Cluster"), not a part of the lifecycle

30

Teams and groups



31

Organization

Cluster-based, not process-based

- Team: a few students (2 or 3) from one university
- Group: collection of teams

Each group does the full project

Each team does a part of the project

This is a part of the system ("Cluster"), not a part of the lifecycle

32

Example system & clusters (Fall 2007 course)

Analyze call for papers to feed Informatics Europe's Computer Science Event List (CSEL)

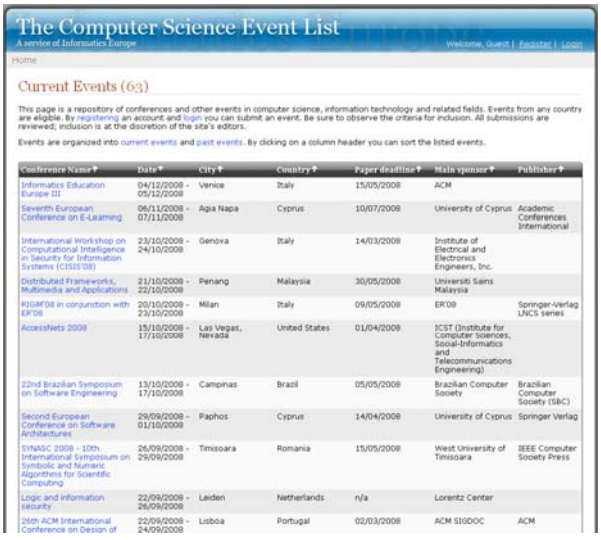
Clusters:

- Natural language analysis
- Editor (for human correction)
- Orchestration, and connection with CSEL

33

The Computer Science Events List

events.informatics-europe.org



34

Example system & clusters (Fall 2007 course)

Analyze call for papers to feed Informatics Europe's Computer Science Event List (CSEL)

Clusters:

- Natural language analysis
- Editor (for human correction)
- Orchestration, and connection with CSEL

35

Main lesson from first session

The importance of APIs

Techniques of abstraction & contracts

36

Lessons from first session

- Success (partial)! All 3 consortiums did reasonably well, one produced a working implementation
- It took a few weeks for students to understand what this was about, and some more to really get to work
- Each consortium found ways to interact, e.g. regular Skype meetings, in spite of often bad conditions
- All found the project hard
- All found it extremely enriching, learned a lot
- Worked much more than expected for such a course
- Understood the great, great, great importance of APIs

37

Project presentation

Attended by students from all universities involved

(through Skype)

38

Next time

Fall 2007 (18 September to 18 December)

More universities will join (Politecnico di Milano, …)
Learn from previous session, develop interesting system

What you need:

- Interest and commitment
- A suitable course, with reasonably similar schedule
- A group of potentially interested students
  (may be volunteers in an existing course)

Use of Eiffel is not an issue (but is a benefit)

39

Join us!

se.ethz.ch/dose

Or just write to me

Bertrand.Meyer@inf.ethz.ch

40

Other innovative teaching work at ETH

Rethought & overhauled the intro prog. course

"Outside-in" approach (inverted curriculum)

- Learn from example
- Thoroughly O-O (incl. inheritance, genericity etc.), Eiffel
- 150,000-line "Traffic" library: multimedia 3-D simulation
- Use powerful software through interfaces & contracts
- From consumer to producer
- Gentle introduction of formal methods (Design by Contract)
- Outstanding student projects (e.g. games, scripting)
- Online textbook (to be published 2008): ***Touch of Class***

touch.ethz.ch

41

Lessons

The world has gone global, so has the software world

Many difficult issues, failure always possible

Some solutions exist, we can teach them

Many software engineering lessons apply, made even more relevant by distributed development

Communication is the core issue:

- Between people
- Between modules: crucial role of APIs and contracts

Infrastructure (network, tools…) is critical

Universities should teach this

42

Join us!

se.ethz.ch/dose

Or just write to me

Bertrand.Meyer@inf.ethz.ch

43