Two or three things I would like to know (empirically)

Bertrand Meyer

Конференция Сифуд (SEAFOOD)
Санкт-Петербург, Июнь 2010

ETH
Chair of Software Engineering


AJAX
WC
PREMIER PRIX !
20 ans
UN FILM DE
JEAN-LUC GODARD
deux ou trois choses
que je sais d'elle...
RAOUL COUTARD

Supplementary topics

- Experiences in industry and academic distributed development

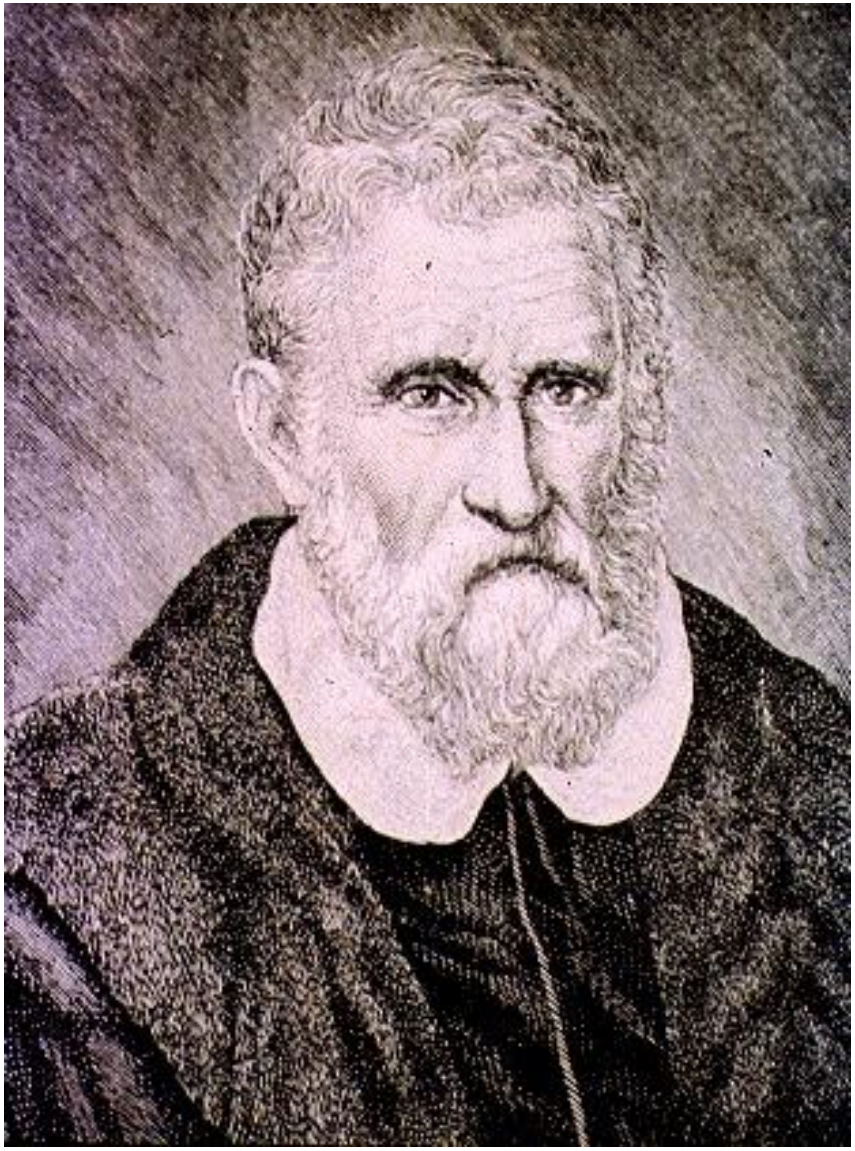
- Verification research at ETH Zurich

Great ideas

Structured programming
Object-oriented programming
Design by Contract
Object-oriented analysis
Seamless development
Test-driven development
Model-driven architecture
UML
Use cases
Pair programming
Refactoring
Scrum
Aspect-oriented programming

How do we know they work?

The Marco Polo principle (R. Lister)



"I traveled far and saw wonderful things"

Example statement (Dijkstra, 1968)

*"For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such disastrous effects, and I became convinced that the **go to** statement should be abolished from all "higher level" programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so."*

Another example: the Agile manifesto

Manifesto for Agile Software Development

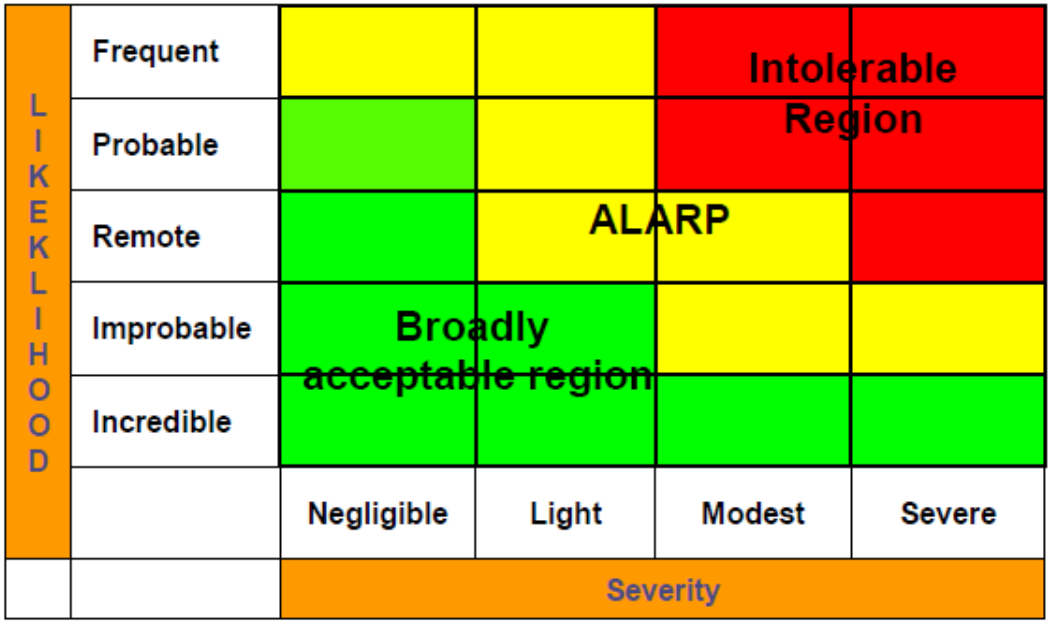
We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

How the rest of the world views software



ALARP = As Low As Reasonable Practical

Software (IEC 62304):

LIKELIHOOD = 100%

ISO 14971 (medical devices):

Risk = f (*LIKELIHOOD*, *Severity*)

Source: C. Gerber, Stryker Navigation

What the field needs

Two complementary views:

- Deductive:
  "Try my approach!"

- Inductive:
  "I tried this and it
  - *Worked!*
  - *Didn't work!"*

Cf physics:

- Theoretical
- Experimental

A horror story

Semicolon as:

- Separator (Algol):

  p ; q ; r

  -- As in: f (x, y, z)

- Terminator (C):

  p ; q; r;

Why do Ada, C++, Java, C#... use terminator convention?

Wrong!

- **Syntax errors only**
- **PL/I-trained programmers**
- **In separator language, extra semicolon is error!**

Answer: Gannon & Horning, *Language Design for Programming Reliability*, IEEE Trans. on S.E., June 1975

Experiment: programmers in language with terminator convention make fewer mistakes

The mistakes that happen in practice

```
while (e) ; a
```

```
if (e) then
        a
else ;
        b
```

A horror story

Semicolon as:

- Separator (Algol):

  p ; q ; r

- Terminator (C):

  p ; q; r;

-- As in: f (x, y, z)

Why do Ada, C++, Java, C#... use terminator convention?

Wrong!

- **Syntax errors only**
- **PL/I-trained programmers**
- **In separator language, extra semicolon is error!**

Answer: Gannon & Horning, *Language Design for Programming Reliability*, IEEE Trans. on S.E., June 1975

Experiment: programmers in language with terminator convention make fewer mistakes

Empirical software engineering

Advocated for many years by such people as Barry Boehm, Vic Basili, Watts Humphrey, Walter Tichy, Andreas Zeller, …

Aim: subject software engineering claims to rigorous experimental evaluation

Many more papers recently: ICSE, ESEC, ESEM

By the way…

LASER Summer School on Software Engineering

Empirical Software Engineering

September 5-11, 2010 - Elba Island, Italy

Victor Basili (University of Maryland)
Barry Boehm (University of Southern
Natalia Juristo (Universidad Politécnic
Bertrand Meyer (ETH Zurich, director)
Nachi Nagappan (Microsoft Research)
Walter F. Tichy (University Karlsruhe)

http://se.ethz.ch/laser

Early empirical papers

Industry: not reproducible

University: not credible

What has changed

In the past ten years, the availability of large open-source project repositories has provided empirical software engineering researchers with a wealth of objective material that makes verifiable, repeatable analyses possible

Some commercial software has also become available for examination, e.g. from Microsoft

Simple sample questions

1. Do novice programmers produce more bugs (in Eclipse)?
   (Andreas Zeller)

2. Are more tested modules less bug-ridden?

3. Are goto-rich modules more bug-prone (in Eclipse)?
   (Andreas Zeller)

Empirical SE papers, today

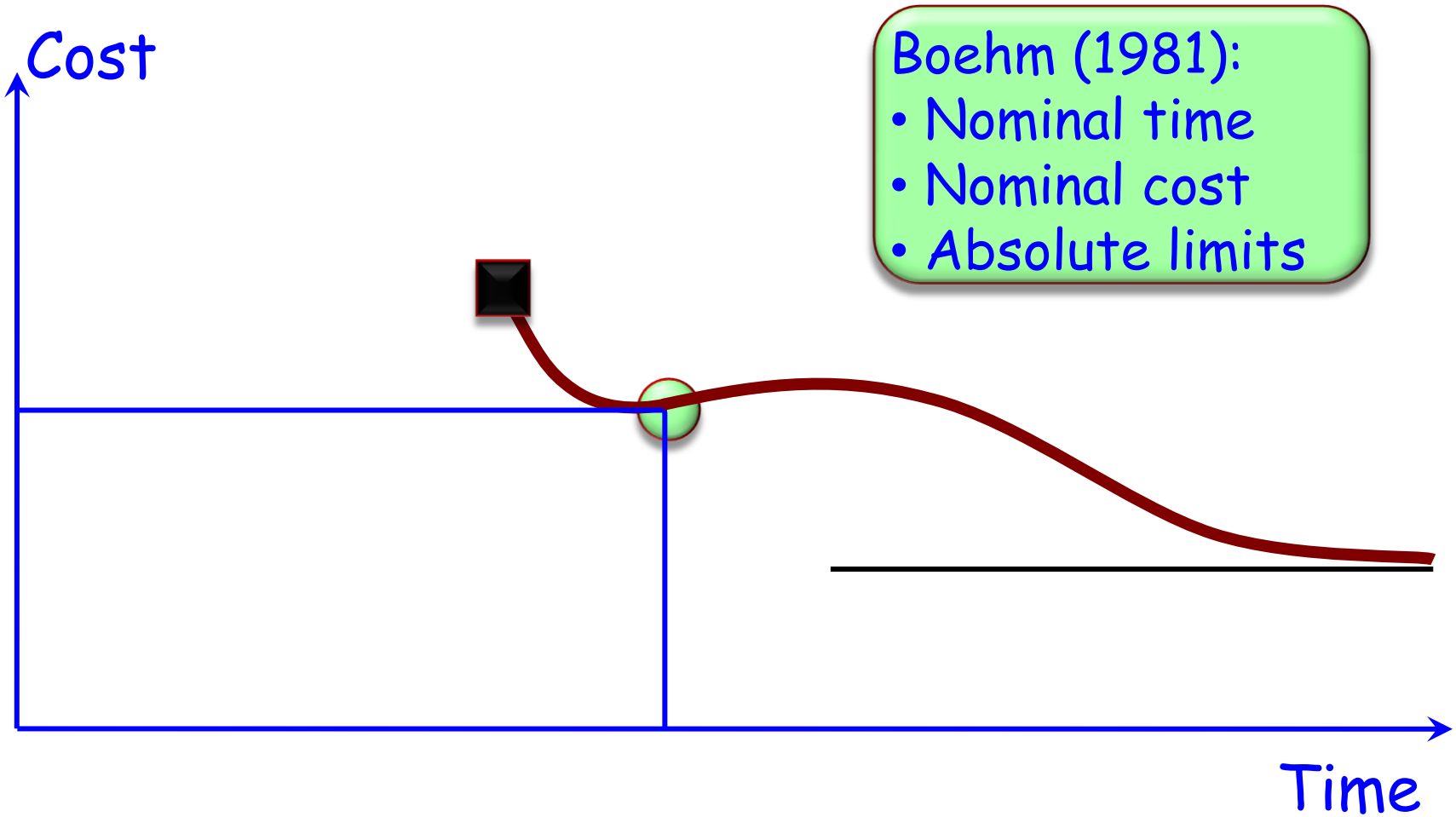
Better than they used to be, but:

- Often very disappointing, e.g. many studies ask people what they think instead of using objective measures
- "**Threats to Validity**" section kills generalization

Sample open questions: pair programming

1. Does it lead to fewer bugs?
2. Does it lead to shorter debugging times?
3. Are there good programmers who will not adapt to it?
4. Should it be applied throughout the programming phase?
5. Should it be applied to other tasks, e.g. pair specifying, pair testing?
6. Are there useful variants, e.g. programmer-tester pairing?

Sample open questions: nominal values



Boehm (1981):
- Nominal time
- Nominal cost
- Absolute limits

Sample open questions: refactoring

What is better:

- Design?
- Refactoring?
- Some combination?

Sample open questions: tests vs specs

What works better:

- Extensive specifications?
- A test-driven process?
- Some combination?

Sample question: RTC vs CTR

Commit strategies:

- Review Then Commit (Google, original Apache)
- Commit To Review (Apache)

See Rigby, German, Storey, *Open Source Software Peer Review Practices: A Case Study of the Apache Server*, ICSE 2008, but need studies on other projects and correlation with software quality measures!

Sample open question: complexity measures

Which measures correlate best to quality indicators?

- SLOC
- Function points
- Specific O-O metrics
- McCabe etc.

Sample open question: testing

When should we stop testing?

Conditions for progress

Better refereeing process

- Experimental work acceptable
- Reproducibility papers acceptable
- "No surprise" dismissal not valid

Openness

- All code and data available on Web
- All assumptions disclosed

Reproducibility

No exaggerated "Threats to Validity" excuses

A plan

Select ten questions

Assemble panel of experts

Publicize questions, invite answers

Publication date: July 2010 (TOOLS)

Submission date: February 2011

Workshop: July 2011 (TOOLS)

Supplementary topics

- Experiences in industry and academic distributed development

- Verification research at ETH Zurich

Verification research at ETH Zurich

Our verification research

Automatic testing: AutoTest

- Manual testing (called "automatic testing" elsewhere, e.g. Junit)
- Test generation
  - No manual test suites or test cases
  - No oracles (they come from the existing contracts)
  - Push-button
- Test extraction: generate reproducible test cases from failures

Automatic bug fixing: AutoFix

Full specifications: EiffelBase 2

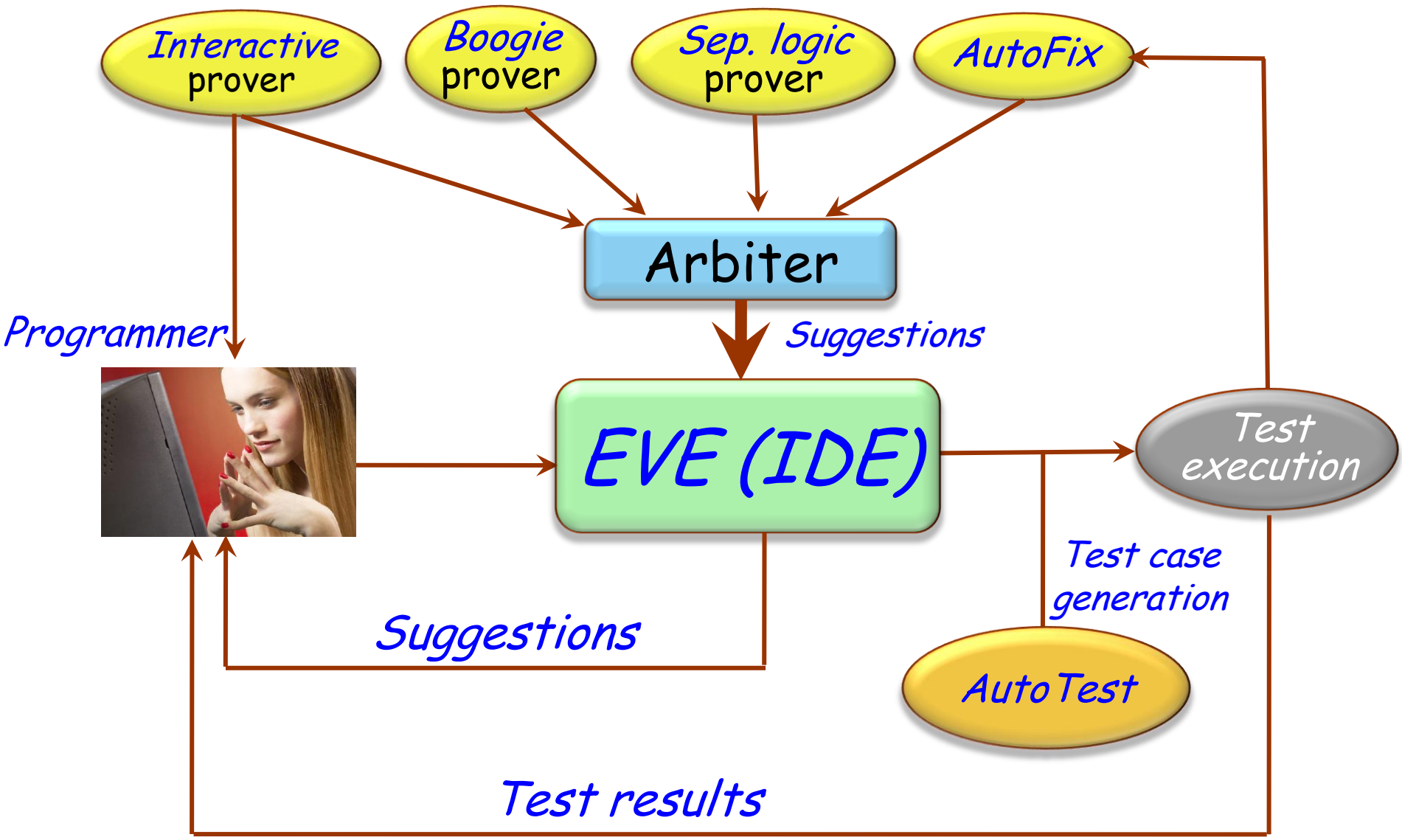
Proofs: Hoare-based

Proofs: Object-oriented programs (the alias calculus)

Proofs: Separation logic

Proofs and tests: concurrency (SCOOP)

VAMOC: Verification As A Matter Of Course



Not shown but important

- Invariant generation
  (Carlo Furia)
- Full contracts
  (Nadia Polikarpova)
- Proof transformation
  (Martin Nordio)
- Fix suggestions
  (Yi Wei, Yu Pei, joint work with Andreas Zeller)

What makes it all possible

Contracts throughout

Try our techniques:

- http://eiffel.com
- http://se.ethz.ch

Experiences in academic & industry software development

Distributed Software Development

Two case studies, lessons and challenges:

- Industry: experience with distributed development at Eiffel Software
- Academia: the distributed course project (DOSE) at ETH Zurich

EiffelStudio development

Eiffel Software, in Santa Barbara (Calif.), since 1985

Two-million line code base (almost all Eiffel, a bit of C)

Major industry customers, mission-critical applications

Open-source license, same code, vigilant user community

6-month release schedule since 2006

My role: more active in past two years

Developer group ecosystem:

- 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

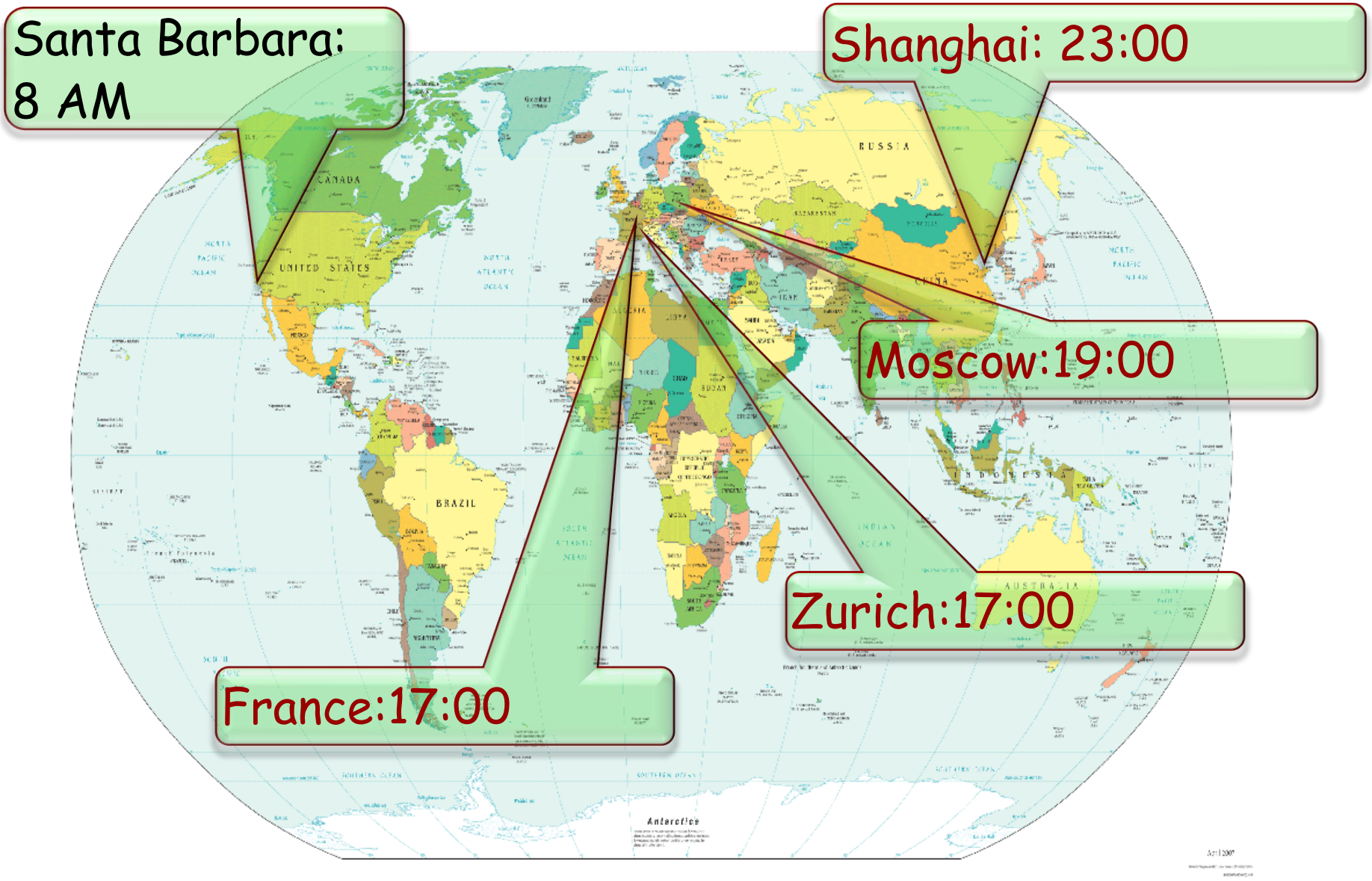
Principle

Every team needs a regular meeting

Our solution: the weekly one-hour meeting

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

How do we organize a meeting?



Meeting tools: now

Webex for conference call management

(Used X-Lite, gave up)

Google Docs

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

Skype: chat window only

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

Humans can multiplex!

Time is strictly limited: one hour

ISE weekly meeting

Messages in this chat are older than 3 hours

[redacted] says: 21-Feb-08 17:00:45
Good morning/evening

[redacted] says: 21-Feb-08 17:00:46
Hello

[redacted] says: 21-Feb-08 17:04:21
For info: the doc's url as preview:
http://docs.google.com/View?revision=_latest&docid=dd7kn5vj_8gmxzhffv&hl=en

[redacted] says: 21-Feb-08 17:05:28
there is an echo

21-Feb-08 17:05:48
never mind

[redacted]) says: 21-Feb-08 17:17:50
I disagree.
When there is a crash, if the we have multi lines, then we can know exact error point. If we write them in one line, then we have to guess.

[redacted] says: 21-Feb-08 17:18:45
we need to improve the RTNHOOK macro

21-Feb-08 17:18:55
if we improve it that it won't be a problem

21-Feb-08 17:19:00
RTHOOK (1)

[redacted]s: 21-Feb-08 17:19:10
we have bp slot index, .. we would need to show the "nested bp slot index"

21-Feb-08 17:19:17
that's possible ... somehow

[redacted] says: 21-Feb-08 17:19:26
RTNHOOK (1,1); /* First instruction, first nested or expression */

[redacted] says: 21-Feb-08 17:20:15
Ok if we have the `nested bp slot index' issue.

21-Feb-08 17:20:26
Ok if we have the `nested bp slot index' feature.

[redacted]ays: 21-Feb-08 17:21:48
It's possible to view expressions in the debugger.

[redacted] says: 21-Feb-08 17:27:14
indeed sometime doing the evaluation is not desired (due to potential side effects)

[redacted] says: 21-Feb-08 17:28:19
I was just curious of clear rules about IEK.5.1

[redacted]
[redacted]
[redacted]
[redacted]
Eiffe
[redacted]
mist
[redacted]
[redacted]

Principles

Scripta manent:
Organize meetings around shared documents

Code review

Traditional: time-consuming, tedious, value often questioned as compared to e.g. static analysis tools

With the Web it becomes much more interesting!

- Classes circulated three weeks in advance
- Comment categories: choice of abstractions, other aspects of 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 is devoted to unresolved issues

Goal of the DOSE course at ETH Zurich

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
- …

Project: involving other universities

Since 2007:

- Odessa National Polytechnic (Ukraine)
- University of Nizhny Novgorod (Russia)
- Politecnico di Milano (Italy) (C. Ghezzi & E. di Nitto)
- University of Debrecen (Hungary)
- University of Zurich
- Hanoi University of Technology (Vietnam)
- (2010) University of Rio Cuarto (Argentina)

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

Teams and groups

	Group 1		Group 2		Group 3	
University A	Team A1	Team A2		Team A3		Team A4
University B	Team B1		Team B2	Team B3		
University C	Team C1		Team C2	Team C3		
University D	Team D1		Team D2			
University E	Team E1	Team E2		Team E3		Team E4

DOSE 2007 project results

- Delays to set up the projects
- Lack of communication
  - Delay in replying to e-mails
  - Technical problems with skype conferences
- Misunderstandings in SRS
- Weak API design
  - Incomplete
  - Ambiguous
- Integration partially failed

Software Requirements Specification

D.1. The system shall be able to extract the elements of a call for paper from text e-mails.

D.2. The system can send the e-mail only if at least all key elements have been extracted or introduced by the user. The key elements are: *(1) conference name, (2) conference dates, (3) abstract and submission deadline, (4) conference category, and (5) URL of the conference.*

D.3. The conference category is either *"Conference"* *or* *"Symposium"* *or* *"Workshop"* *or* *"Summer School"*

Some problems

Case 1 - Submission deadline:

- Team A: *day.month.year*
- Team B: integers for the day and year but a string (such as "January" or "February") for the month.

Case 2 – Abstract deadline earlier than submission deadline:

- Team A: Not checked
- Team B: Checked – Exceptions were triggered

Solution: class specification

```
class EVENT feature
  submit_to_csel
        -- Submit the conference information by sending an e-mail.
    require
      valid_deadlines: abstract_deadline.earlier_than (paper_deadline)
   do … end
feature -- Implementation
  name: STRING
  abstract_deadline, paper_deadline: DATE
  category: CATEGORY
invariant
        category_status: category.is_conference xor
        category.is_symposium xor
        category.is_workshop   xor
        category.is_summer_school
end
```

Interface: class CATEGORY

```
class CATEGORY  feature -- Status report
    is_conference: BOOLEAN
            -- Does this category represent conferences?
        do end
    is_symposium: BOOLEAN
            -- Does this category represent symposiums?
        do end
    is_workshop: BOOLEAN
            -- Does this category represent workshops?
        do end
    is_summer_school: BOOLEAN
            -- Does this category represent summer schools?
        do end
end
```

Main lesson from first session

APIs are critical

Techniques of abstraction & contracts

DOSE 2008 results

The systems were integrated and the three clusters worked in the same system

Contracts helped to document and understand the interfaces

Contracts in SRS were useful to avoid misunderstandings and to specify the interaction between subsystems

Difficulties (e-mails)

Some members of our team suffer from weak-English

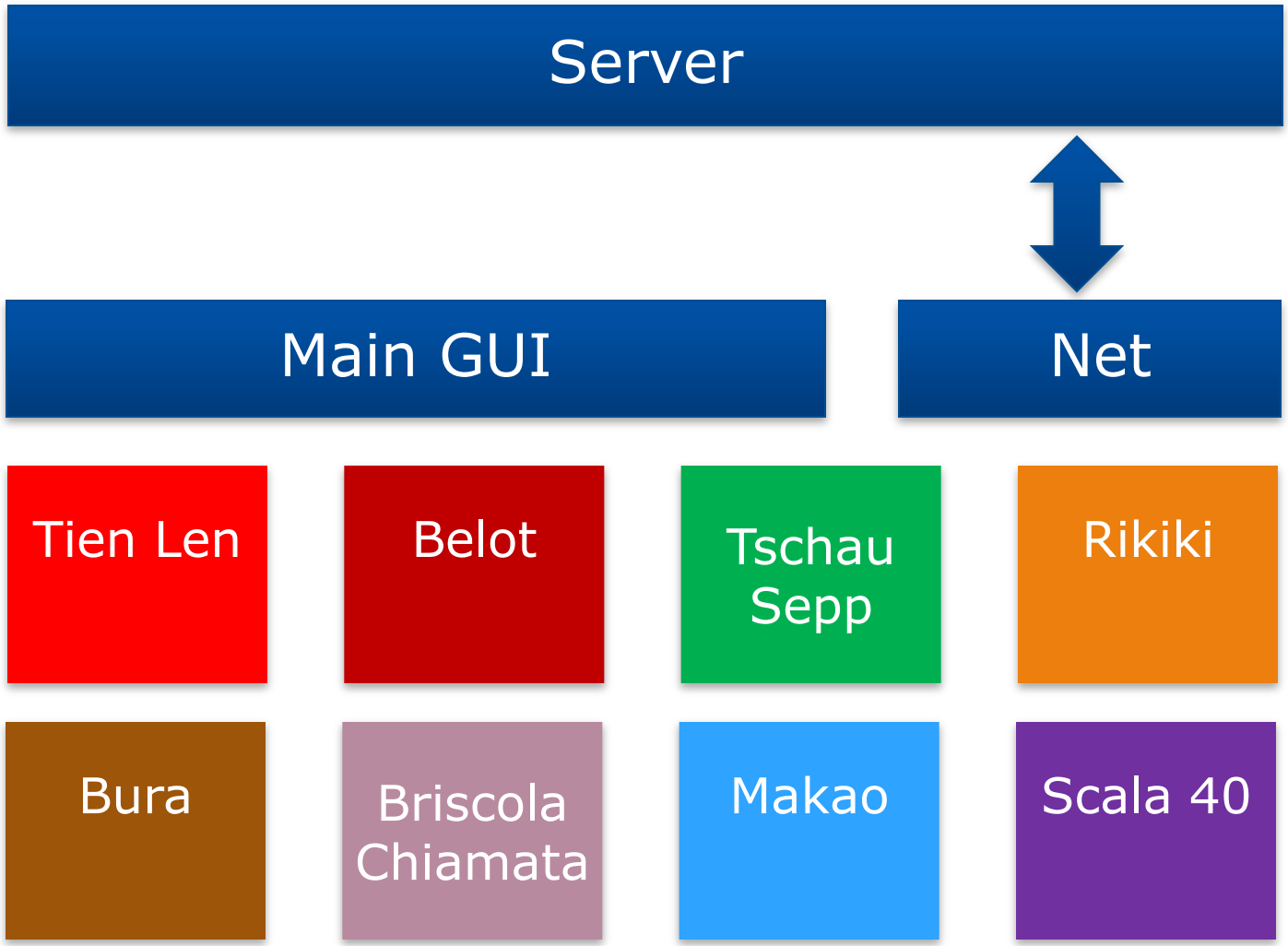
Team A has implemented the system in Java, and we have implemented in Eiffel; now, we cannot integrate it, any hints?

Their document is clearly not consistent with the decisions we took in our last meeting

I'm sorry I could not make it to the implementation meeting yesterday. A water pipe in my apartment burst ... After some frantic hours of fixing and cleaning up, it is now more or less OK

Aleksey couldn't read any emails last week because his Internet cable had been stolen by a drunken bear

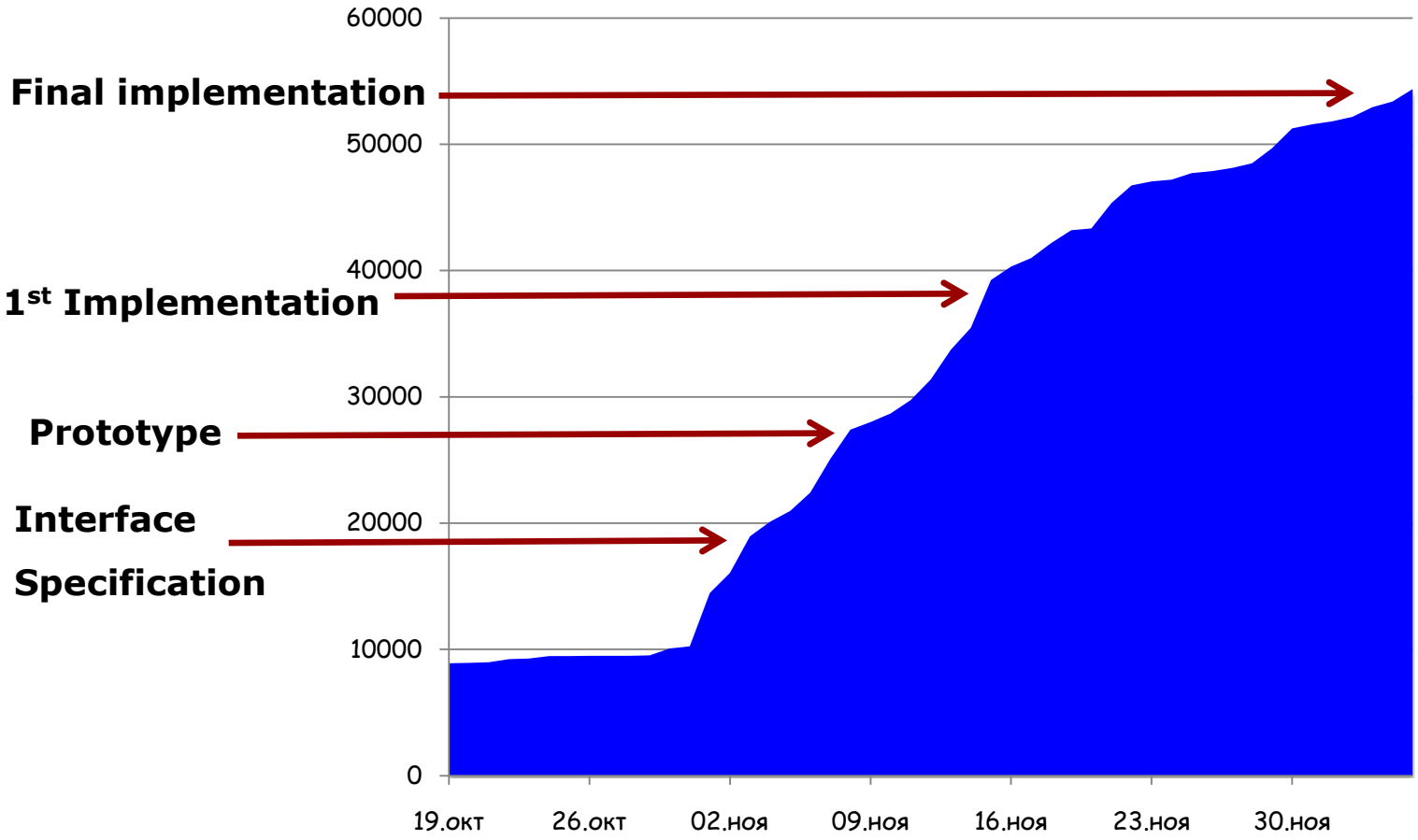
Application Architecture (DOSE 2009)



DOSE 2009 results

8 games fully implemented, integrated and deployed

55'000 lines of code



We are doing it again!

September-December 2010

ICSE SCORE competition

http://se.ethz.ch/dose

Join us!

Final thoughts

Software is special and not: do

Do not overestimate, and do not underestimate, the differences

Not special: it is the engineering of products, based on mathematics

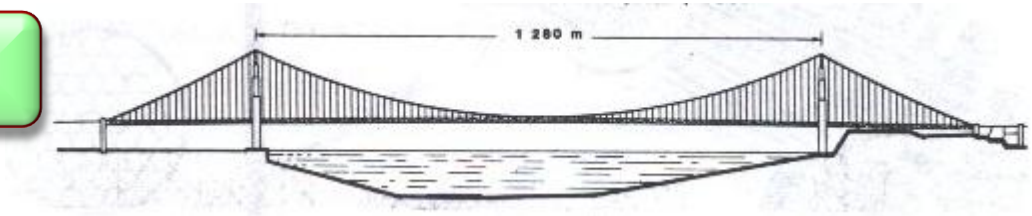
Special:

- Virtual product

  "The industry of pure ideas"
- Design only, no production
- No degradation
- Complexity
- Change
- Description-Implementation Porosity

Description and implementation

The Drawing of the Bridge



The Bridge

Is this a program?

Specification (VDM)

```
AccNum = token;
CustNum = token;
Balance = int;
Overdraft = nat;
AccData :: owner : CustNum
        balance : Balance
state Bank of
        accountMap : map AccNum to AccData
        overdraftMap : map CustNum to Overdraft
inv mk_Bank(accountMap,overdraftMap) ==
        for all a in set rng accountMap & a.owner in set
                dom overdraftMap and
                a.balance >= -overdraftMap(a.owner)
```

Is this a program?

```
note
        description :
  "Individual fragments of a schedule"
deferred class SEGMENT feature
        schedule : SCHEDULE deferred
end
        -- Schedule to which
        -- segment belongs
        index: INTEGER deferred end
        -- Position of segment in
        -- its schedule
        starting_time, ending_time :
                INTEGER deferred
end
        -- Beginning and end of
        -- scheduled air time
        next: SEGMENT deferred end
        -- Segment to be played
        -- next, if any
```

```
sponsor : COMPANY deferred end
        -- Segment's principal sponsor

rating : INTEGER deferred end
        -- Segment's rating (for
        -- children's viewing etc.)

    ... Commands such as change_next,
    set_sponsor, set_rating omitted ...

Minimum_duration : INTEGER = 30
    -- Minimum length of segments,
    -- in seconds

Maximum_interval : INTEGER = 2
    -- Maximum time between two
    -- successive segments, in seconds
```

Is this a program?

```
invariant
        in_list: (1 <= index) and (index <= schedule.segments.count)
        in_schedule: schedule.segments.item (index) = Current
        next_in_list: (next /= Void ) implies
                (schedule.segments.item (index + 1) = next)
        no_next_iff_last: (next = Void) = (index =
schedule.segments.count)
        non_negative_rating: rating >= 0
        positive_times: (starting_time > 0 ) and (ending_time > 0)
        sufficient_duration:
        ending_time – starting_time >= Minimum_duration
        decent_interval :
        (next.starting_time) - ending_time <= Maximum_interval
end
```

Commercial

```
note
        description: "Advertizing segment"
deferred class COMMERCIAL inherit
        SEGMENT
                rename sponsor as
advertizer end
feature
        primary: PROGRAM deferred
        -- Program to which this
        -- commercial is attached
        primary_index: INTEGER deferred
        -- Index of primary
```

```
set_primary (p: PROGRAM)
        -- Attach commercial to p.
    require
        program_exists: p /= Void
        same_schedule: p.schedule = schedule
        before:
            p.starting_time <= starting_time
    deferred
    ensure
        index_updated:
            primary_index = p.index
        primary_updated: primary = p
    end
```

```
invariant
    meaningful_primary_index: primary_index = primary.index
    primary_before: primary.starting_time <= starting_time
    acceptable_sponsor: advertizer.compatible (primary.sponsor)
    acceptable_rating: rating <= primary.rating
end
```

Description-Implementation Porosity

Models and programs

To program is to understand

(Kristen Nygaard)

Seamless development (Eiffel)

The Single Product Principle:

The program is the model

The model is the program

Great ideas

Structured programming
Object-oriented programming
Design by Contract
Object-oriented analysis
Seamless development
Test-driven development
Model-driven architecture
UML
Use cases
Pair programming
Refactoring
Scrum
Aspect-oriented programming

How do we know they work?