Software Trajectory Analysis

an empirically-based method for software process discovery

Pavel Senin

University of Hawaii at Manoa

Department of Information and Computer Sciences

Collaborative Software Development Laboratory

http://csdl.ics.hawaii.edu

senin@hawaii.edu

Questions for today

- The programming: what is this exactly about?
- What are software processes and why do I care? Moreover, why do I think you should care?
- What is going on in the field? Which is the best software process?
- What I do
  - Research hypothesis
  - Methods
  - what is the planned contribution of my thesis?

Invention of the Programming

http://en.wikipedia.org/wiki/Ada_Lovelace

Augusta Ada King,
Countess of Lovelace
(10 December 1815 – 27 November 1852)
born **Augusta Ada Byron**
the daughter of a marriage between the
poet George Gordon Byron and Anne Isabelle Milbanke

. . . It may be desirable to explain, that **by the word operation, we mean any process which alters the mutual relation of two or more things,**
be this relation of what kind it may.
This is the most general definition, and would include all subjects in the universe . . .

. . . Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations,
the **engine (Analytical) might compose elaborate and scientific pieces of music of any degree**
of complexity or extent . . .

Programming is...

PHIL ANSCHUTZ: BUILDING HIS FOURTH EMPIRE
NOVEMBER 8 • 2010 EDITION
Forbes
PAUL GRAHAM
Y COMBINATOR REINVENTS HOW STARTUPS ARE CREATED
THE BEST SMALL

Painting?

...Hacking and painting have a lot in common. In fact, **of all the different types of people I've known, hackers and painters are among the most alike.** What hackers and painters have in common is that they're both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is **make good things...**

or Rock Climbing? (Alistair Cockburn)

It is Technical, Individual and Team activity, we use Tools, do Planning and Improvising, it's Fun, Challenging , Resource-limited, Dangerous...

-to reach the summit.

-to make it easier for subsequent teams to reach the summit.

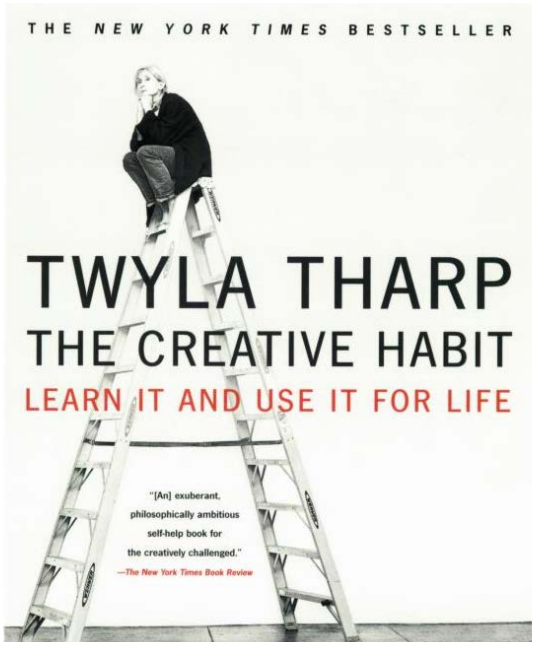
"When I'm writing poetry, it feels like the center of my thinking is in a particular place, and when I'm writing code the center of my thinking feels in the same kind of place."

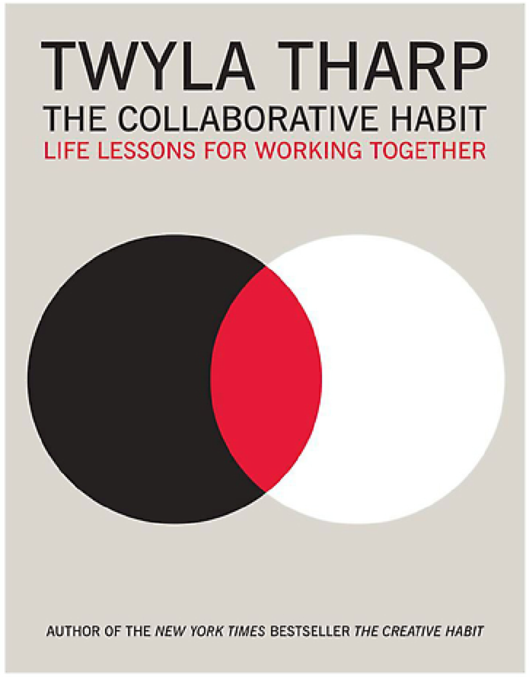
- Richard Gabriel, Distinguished Engineer at Sun Microsystems

Poetry?

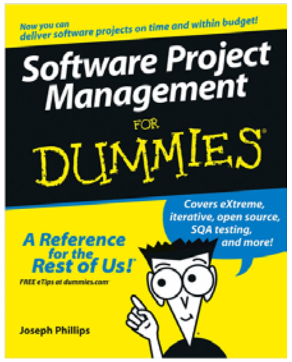
Programming

- It is a creative, human-driven activity, such as choreography (in my opinion)
- As in any other creative activity there are many ways to get things done. With many languages, cool tricks and all other stuff.
  Look at StackOverflow – almost no questions have a single answer. There are comments and long discussions…
- Nevertheless, results are usually deterministic, i.e. you one is right or wrong, since the argument – the code works or not.
- It requires continuous education, not just training, but continuous acquisition of new skills and everyday practice. Like dancing and choreography.


THE NEW YORK TIMES BESTSELLER
TWYLA THARP
THE CREATIVE HABIT
LEARN IT AND USE IT FOR LIFE
"[An] exuberant, philosophically ambitious self-help book for the creatively challenged."
—The New York Times Book Review


TWYLA THARP
THE COLLABORATIVE HABIT
LIFE LESSONS FOR WORKING TOGETHER
AUTHOR OF THE NEW YORK TIMES BESTSELLER THE CREATIVE HABIT

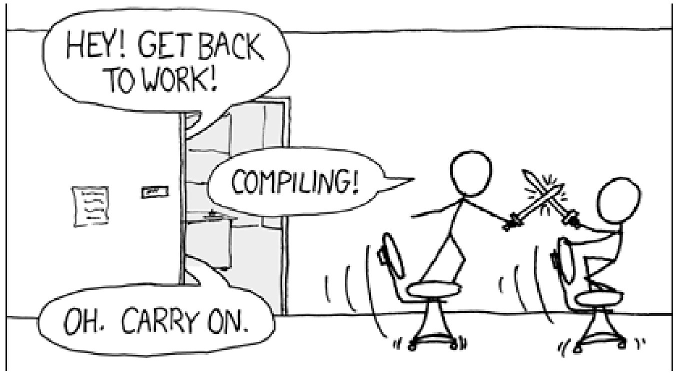
Programmer


Now you can deliver software projects on time and within budget!
Software Project Management FOR DUMMIES
Covers eXtreme, iterative, open source, SQA testing, and more!
A Reference for the Rest of Us!
FREE eTips at dummies.com
Joseph Phillips

- Maybe, it is not a profession, since there is no license given? (i.e. anyone can become a good programmer)
  - ACM abandoned licensing
  - IEEE still has some licensing for "engineers"
  - Yes, in some countries it does exists (Canada, UK)

- Special education is not really needed look at these guys:

- As said by Ada Lovelace, all what we do - is to orchestrate operations over various entities through design, code writing, compiling, testing, debugging, and maintaining the detailed instructions to computers to perform some functions



What depends on these people and their “instructions”?

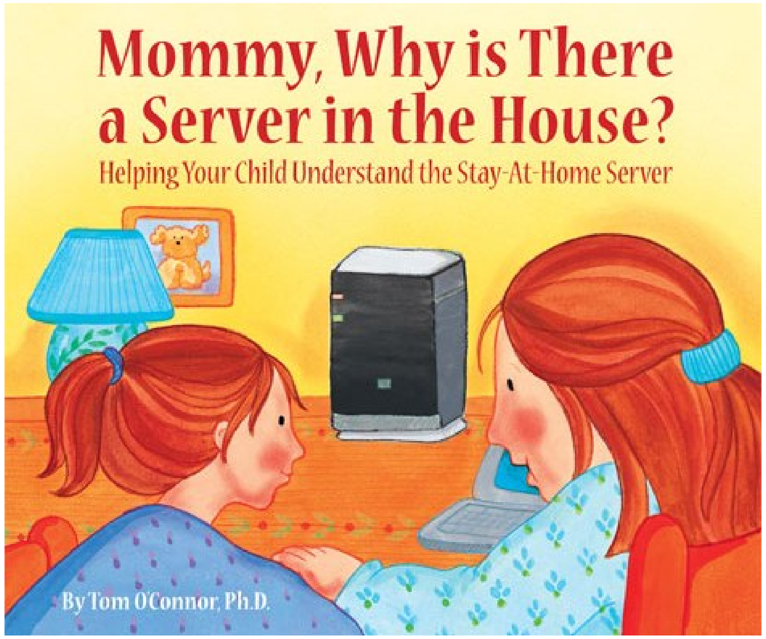
Identity
passport, drivers license, insurance,
health records, digital media

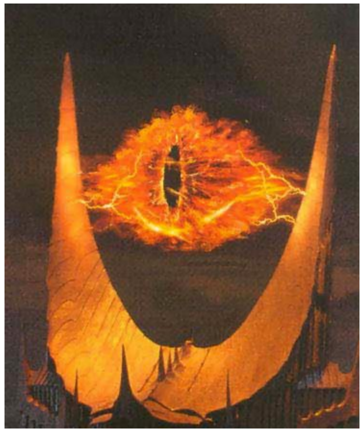
Safety, Transport, and life sustainability
Security and law enforcement, surveillance,
traffic control energy grid, water supply,
food supply, shopping

Money and Business
bank accounts, trading, stocks, financial records
taxes, CB/VISA/PayPal

Communications, Social interactions, Lifestyle, Entertainment
Phones, email, the whole Internet thing, Google,
Facebook, Tumblr...☺

Research →
We are simply unable to do research without CPUs anymore,
… yeah, “those computers”, and Data Centers,
NGS instruments, etc…


Mommy, Why is There a Server in the House?
Helping Your Child Understand the Stay-At-Home Server
By Tom O'Connor, Ph.D.



Bottom line: unlicensed, creative, non-professionals control the way things work ☺
well, while this is not completely true, the opposite is not completely true either....

http://www.codinghorror.com/blog/

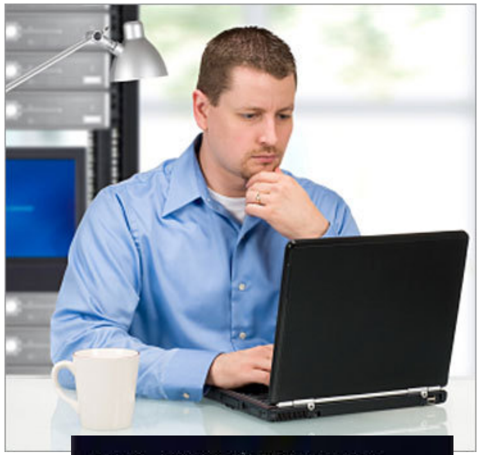
The phenomena was well understood and acknowledged in 1968 at NATO Software Engineering Conference: there the term Software Crisis was coined

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

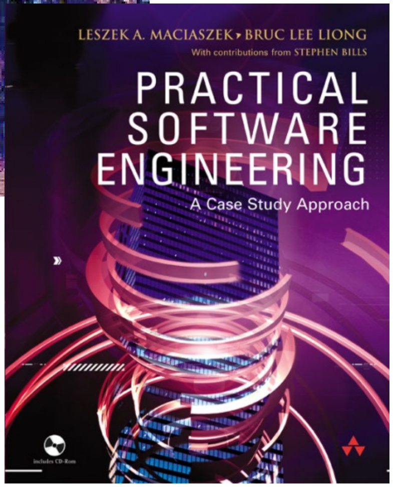
Edsger Dijkstra,
The Humble Programmer
(EWD340), ACM Communications, 1972

Obviously, programming performance doesn't scale as well as CPU does

By analogy to Civil Engineering, the new discipline of Software Engineering was defined to tame the crisis



It defines processes which programmers must follow in order to deliver software .
There is a term ''process conformance''.


LESZEK A. MACIASZEK • BRUC LEE LIONG
With contributions from STEPHEN BILLS
PRACTICAL SOFTWARE ENGINEERING
A Case Study Approach

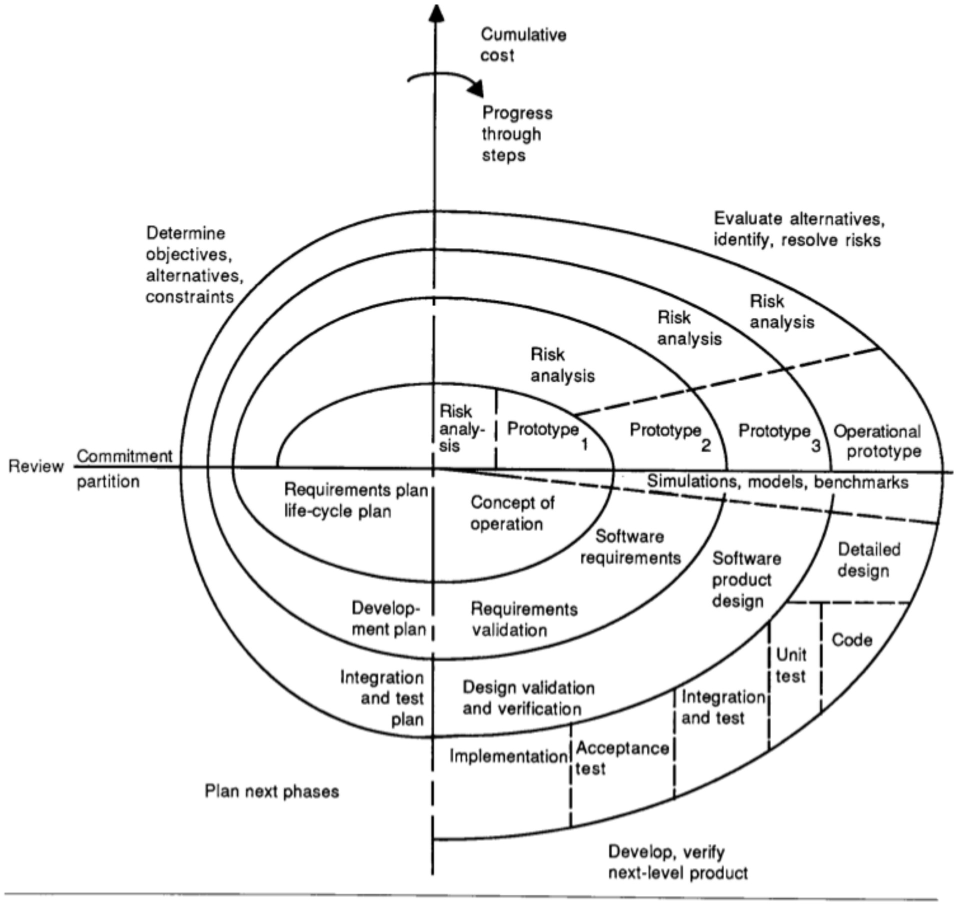


Figure 2. Spiral model of the software process.

...but, quite often, it goes wrong...

Ariane 5 (carrying 4 satellites)

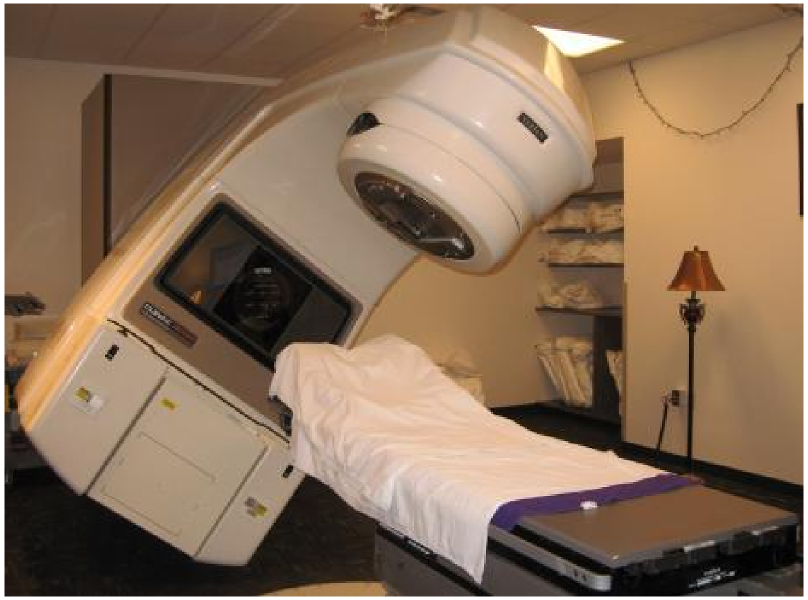
A software bug caused
European Space Agency's Ariane 5 rocket
to crash 40 seconds
into her first flight in 1996
(cost: half billion dollars) 10+ years for credibility recovery

Trustful software from the smaller rocket was used

- Greater horizontal acceleration reading caused overflow exception in conversion from 64-bit floating point to 16-bit signed integer value
- The value was larger than 32,767 - the largest integer storable in a 16 bit signed integer, and thus the conversion failed and an exception was raised
- When the primary computer system failed due to this problem, the secondary system took over.
- The secondary system was running EXACTLY the same software, so it failed EXACTLY the same way!

Therac– 25

patients were given massive overdoses of radiation, approximately 100 times the intended dose causing health damage and loss of life



Interface had a glitch - due to the race condition it allowed technicians, in most cases, bypass the "irritating malfunctions" simply by pressing the "p" key, for "proceed" - beaming 100 times more of radiation. Doing so became a matter of habit...

IRS

Department of the Treasury
Internal Revenue Service

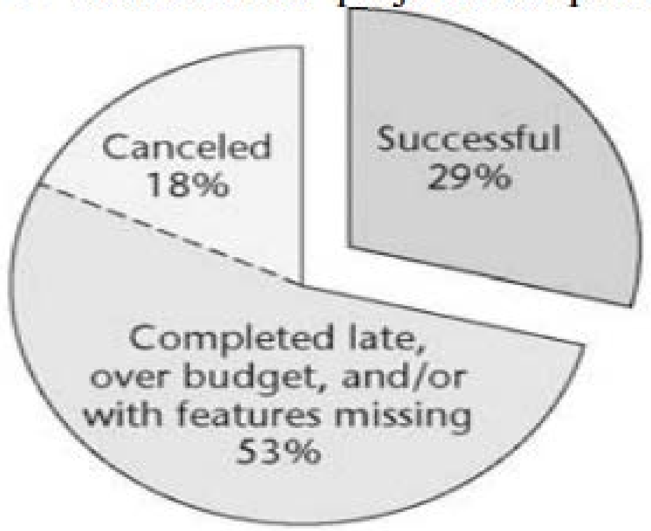
In the early 1980's the IRS hired Sperry to automate tax form processing for $103M By 1985 the cost had tripled, the system could not handle the workload, it had to be replaced...the IRS had to pay interest and overtime wages...

Congresseman Jim Lightfoot called the project ‘a $4-billion fiasco’

40 years later of that NATO Conference

Standish Group Data

- Data on 9236 projects completed in 2004

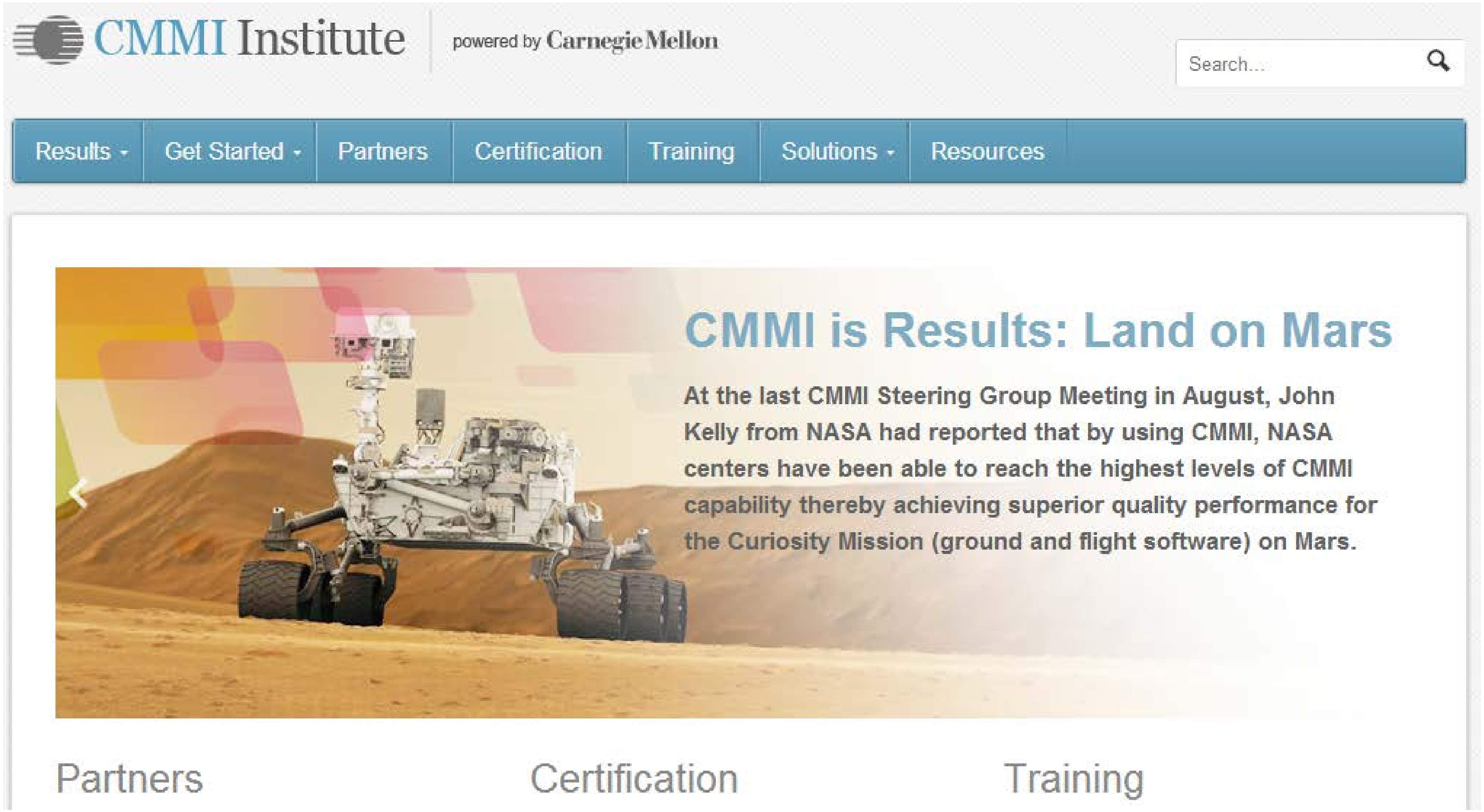


- **More than half of software projects are significantly late, over budget, delivered incomplete or unusable**
- **About quarter of projects simply abandoned without being completed**

... One of the biggest reasons bridges come in on-time, on-budget and do not fall down is because of the extreme detail of design. The design is frozen and the contractor has little flexibility in changing the specifications.
However, in today's fast moving business environment, a frozen design does not accommodate changes in the business practices. Therefore a more flexible model must be used. This could be and has been used as a rationale for development failure...

© The Standish Group, 1994

Engineering: state of the art. It works.



CMMI is Results: Land on Mars

At the last CMMI Steering Group Meeting in August, John Kelly from NASA had reported that by using CMMI, NASA centers have been able to reach the highest levels of CMMI capability thereby achieving superior quality performance for the Curiosity Mission (ground and flight software) on Mars.

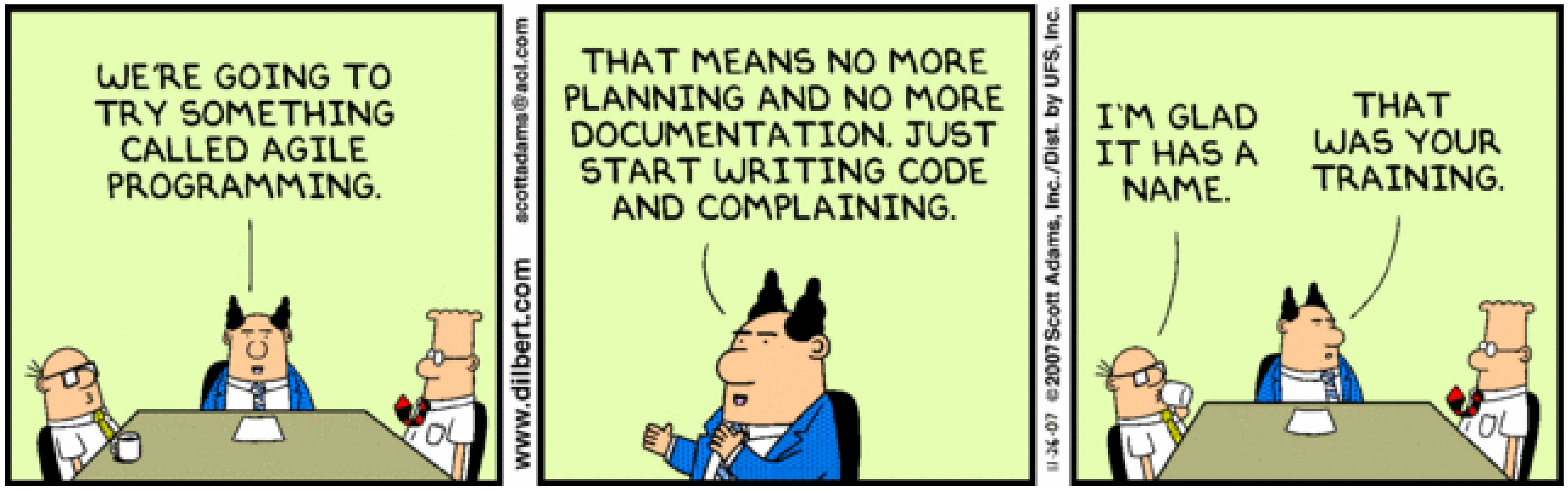
Partners Certification Training

Summary on today’s software engineering landscape

- Uncertainty on “engineering” as a methodology - it might not be _the only_ appropriate way to handle software development complexity.
  - CMMI, ISO, PRINCE2, etc. - all these are excellent products of software engineering evolution.
  - However, while assuring existence of documented processes and proving low variability in delivered projects, they do not guarantee the development and delivery of a good-quality product.
  - The cost of obtaining and maintaining high CMMI levels is prohibitive for majority of software companies.
  - It is thought that Engineering is appropriate only for a narrow area of safety-critical applications with clearly understood requirements which guarantee not to change.
- In reality software projects are way too different from each other. Market is very competitive, resources are scarce, requirements and scope could change on the fly, people leave, new ones arrive, whole teams change. Technology can change, hardware platform can change.
  - It is extremely challenging to establish and to maintain any of large standardized processes in these conditions.

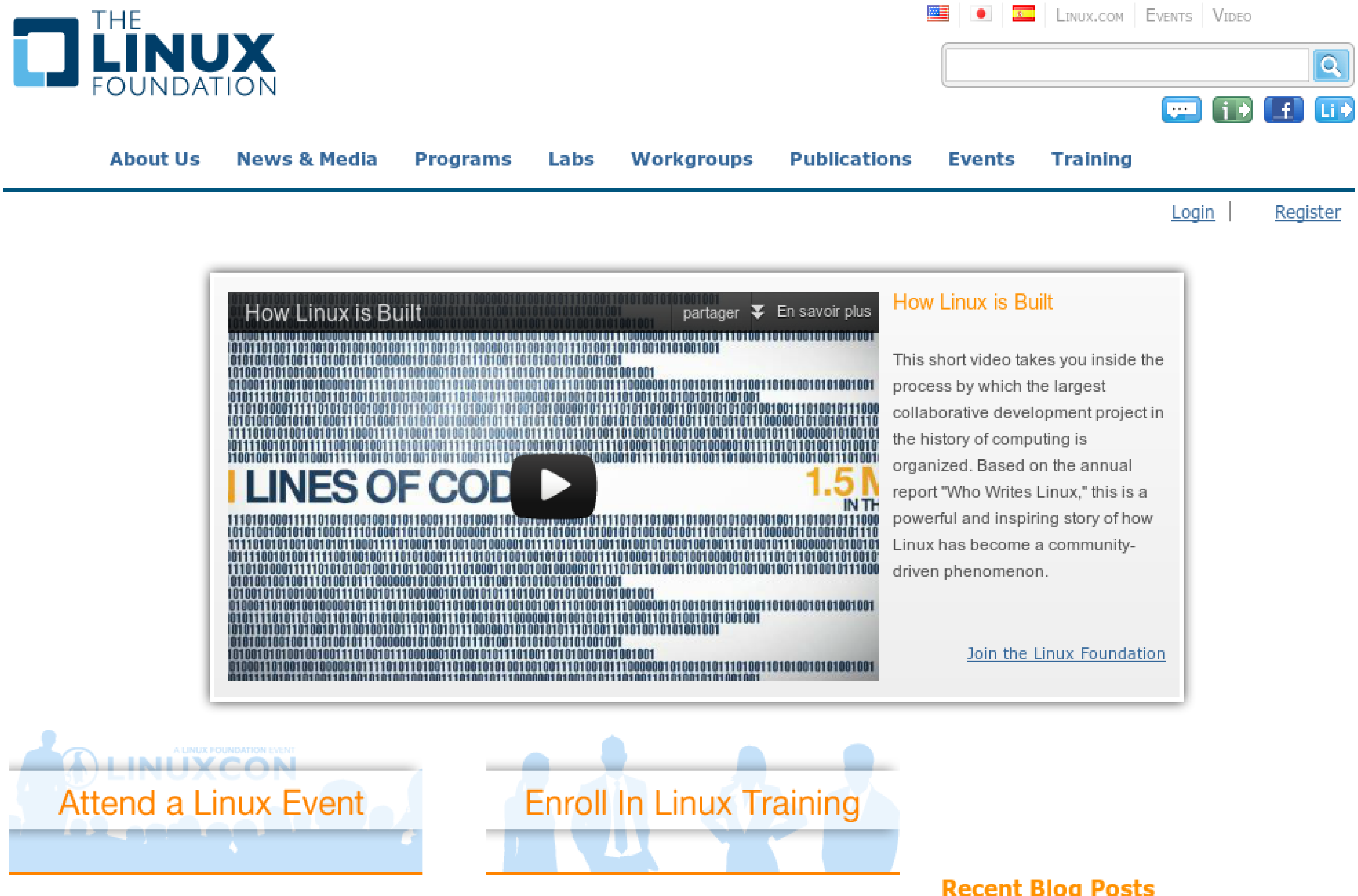
Alternatives to engineering?

- Certainly, there is a problem we have identified in engineering paradigm over years of experience – it does not accommodate the change.
- Which is a problem - it turns out that ability to change is vital for majority of projects.
- Thus, no matter how straightforward, secure, transparent and predictable Engineering is, we must look for more flexible, agile approaches with low up-front expenses and controllable change management.


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
11-26-07 ©2007 Scott Adams, Inc./Dist. by UFS, Inc.

http://universaluclick.com/licensing_permissions/educational_use

Alternatives: the free software model



THE LINUX FOUNDATION

Linux.com | Events | Video

About Us | News & Media | Programs | Labs | Workgroups | Publications | Events | Training

Login | Register

How Linux is Built

This short video takes you inside the process by which the largest collaborative development project in the history of computing is organized. Based on the annual report "Who Writes Linux," this is a powerful and inspiring story of how Linux has become a community-driven phenomenon.

Join the Linux Foundation

Attend a Linux Event

Enroll In Linux Training

Recent Blog Posts

Free software: programmer == user

Date	Thu, 29 Sep 2005 12:57:05 -0700 (PDT)
From	Linus Torvalds <>
Subject	Re: I request inclusion of SAS Transport Layer and AIC-94xx into the kernel...

Again. A "spec" is close to useless. I have _never_ seen a spec that was both big enough to be useful _and_ accurate. And I have seen _lots_ of total crap work that was based on specs. It's _the_ single worst way to write software, because it by definition means that the software was written to match theory, not reality…

So there's two MAJOR reasons to avoid specs:

- they're dangerously wrong. Reality is different, and anybody who thinks specs matter over reality should get out of kernel programming NOW. When reality and specs clash, the spec has zero meaning. Zilch. Nada. None.... It's like real science: if you have a theory that doesn't match experiments, it doesn't matter _how_ much you like that theory. It's wrong. You can use it as an approximation, but you MUST keep in mind that it's an approximation....
- specs have an inevitably tendency to try to introduce abstractions levels and wording and documentation policies that make sense for a written spec. Trying to implement actual code off the spec leads to the code looking and working like CRAP. The classic example of this is the OSI network model protocols. Classic spec-design, which had absolutely _zero_ relevance for the real world.

-....So please don't bother talking about specs. Real standards grow up _despite_ specs, not thanks to them.

-Linus

Manifesto for Software Craftsmanship

Raising the bar.

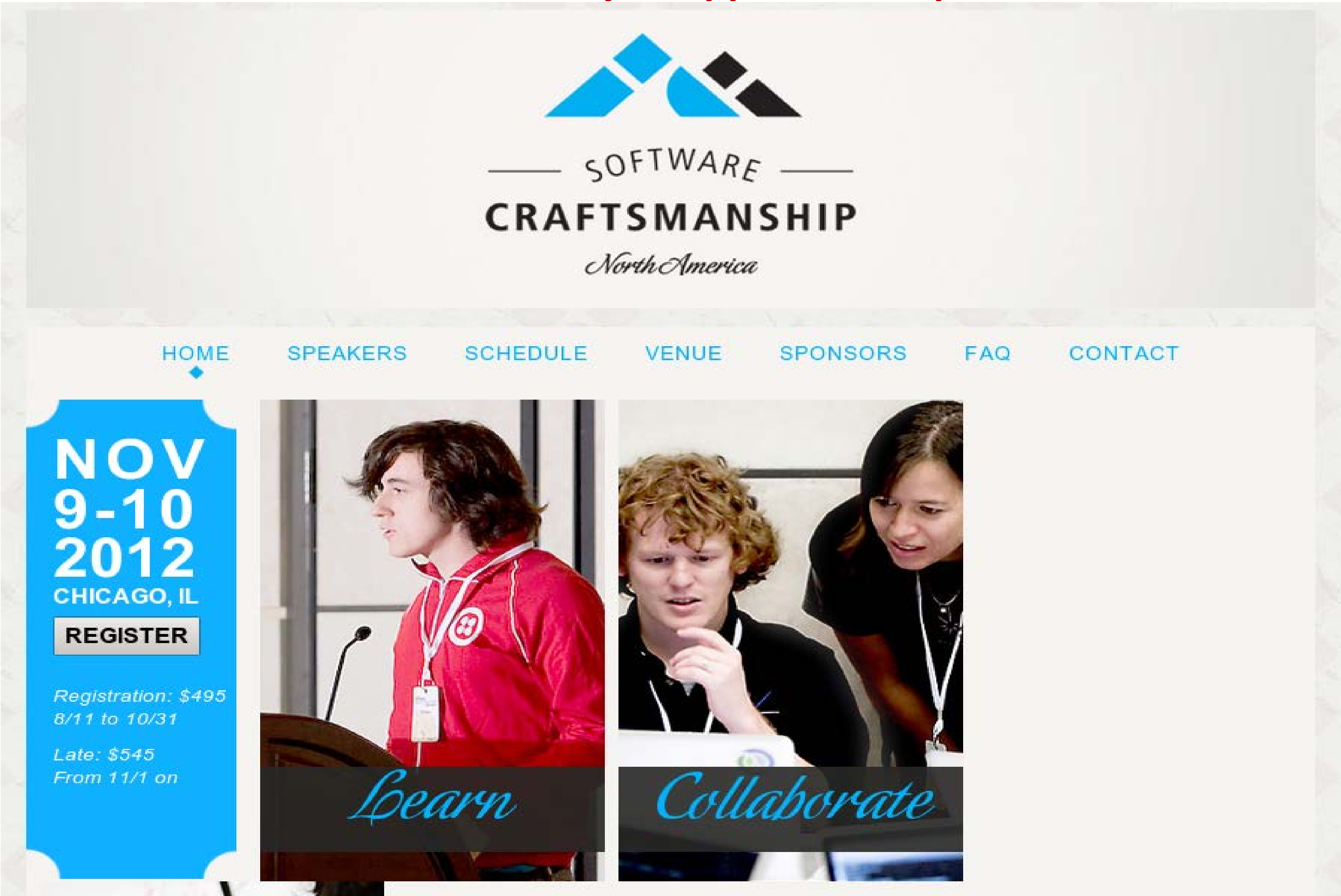
As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

- **Not only working software, but also well-crafted software**
- **Not only responding to change, but also steadily adding value**
- **Not only individuals and interactions, but also a community of professionals**
- **Not only customer collaboration, but also productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

Alternatives: agile programming and craftsmanship

Alternatives: shift to individual processes, personal, craftsman style, Apprenticeship.


SOFTWARE CRAFTSMANSHIP North America

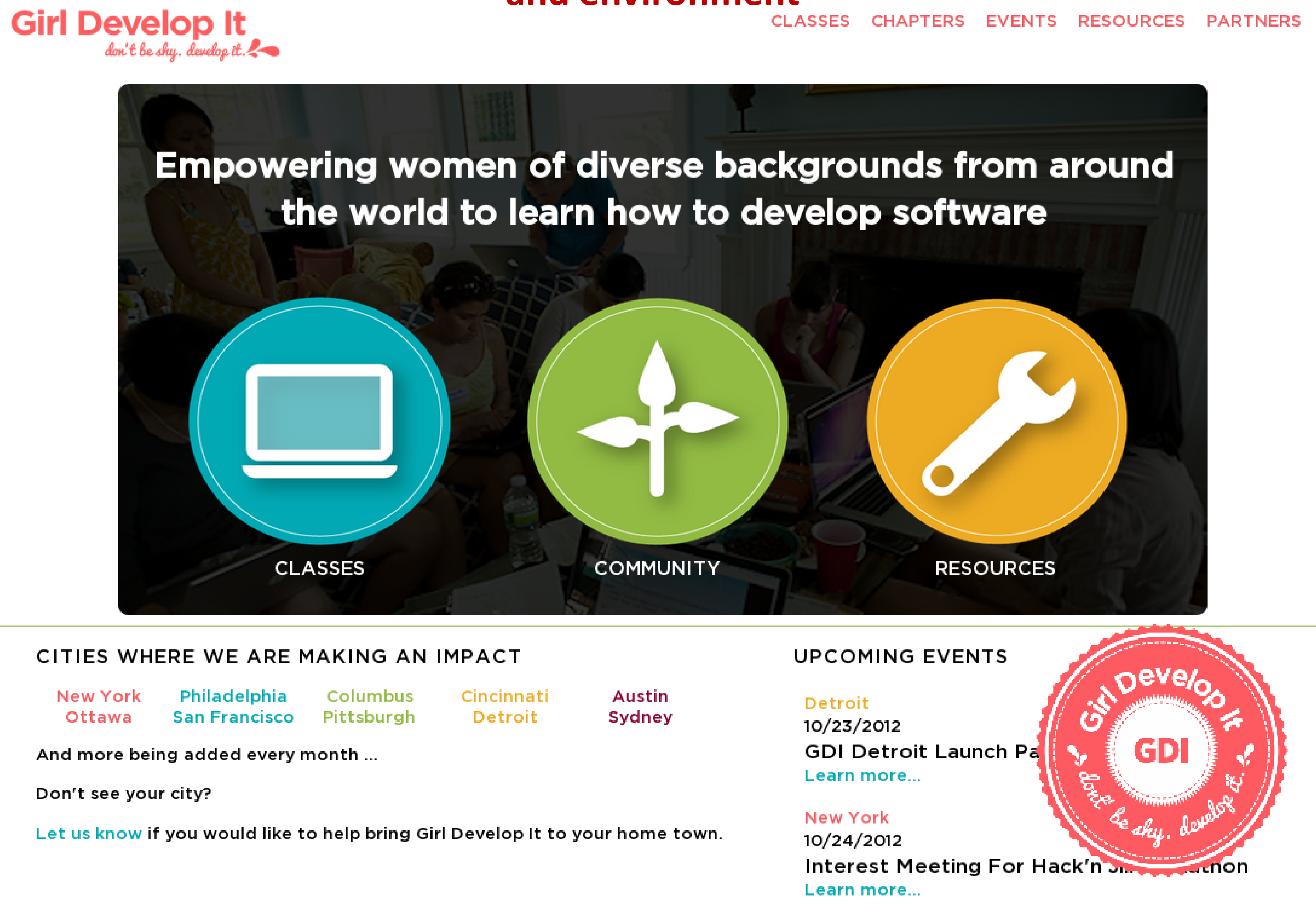
HOME SPEAKERS SCHEDULE VENUE SPONSORS FAQ CONTACT

NOV 9-10 2012
CHICAGO, IL
REGISTER

Registration: $495
8/11 to 10/31

Late: $545
From 11/1 on

Alternatives: Individual pace, very own, gender-specific processes and environment



Empowering women of diverse backgrounds from around the world to learn how to develop software

CITIES WHERE WE ARE MAKING AN IMPACT

New York	Philadelphia	Columbus	Cincinnati	Austin
Ottawa	San Francisco	Pittsburgh	Detroit	Sydney

And more being added every month ...

Don't see your city?

Let us know if you would like to help bring Girl Develop It to your home town.

UPCOMING EVENTS

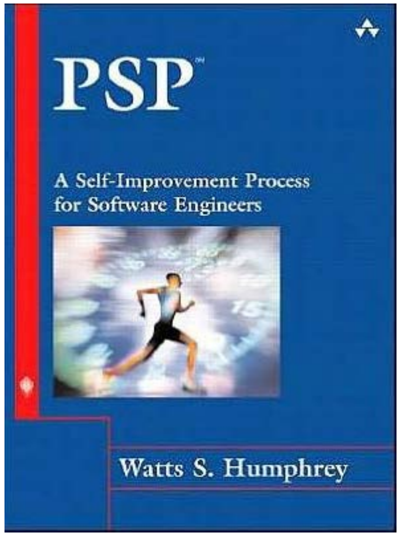
Detroit
10/23/2012
GDI Detroit Launch Pa
Learn more...

New York
10/24/2012
Interest Meeting For Hack'n [illegible]thon
Learn more...

Summary on alternatives to engineering

- All the alternatives emphasize the role of motivated "caring" individuals in creation of high quality software. Role of personal terminal values.
- There is acknowledgement of individual skills, their importance and transfer.
- Concept of continuous product improvement through personal experiences.
- Attention is given to live interactions between programmer and the customer, where care and responsiveness seen as keys to success.

Recently "Software Engineering school" also acknowledged the same:



Software Engineering Institute | Carnegie Mellon

PSP: A Self-Improvement Process for Software Engineers

Personal Software Process (PSP) provides a clear and proven solution. Comprising precise methods developed over many years by Watts S. Humphrey and the Software Engineering Institute (SEI), the PSP has successfully transformed work practices in a wide range of organizations and has already produced some striking results.

http://www.sei.cmu.edu/library/abstracts/books/0321305493.cfm

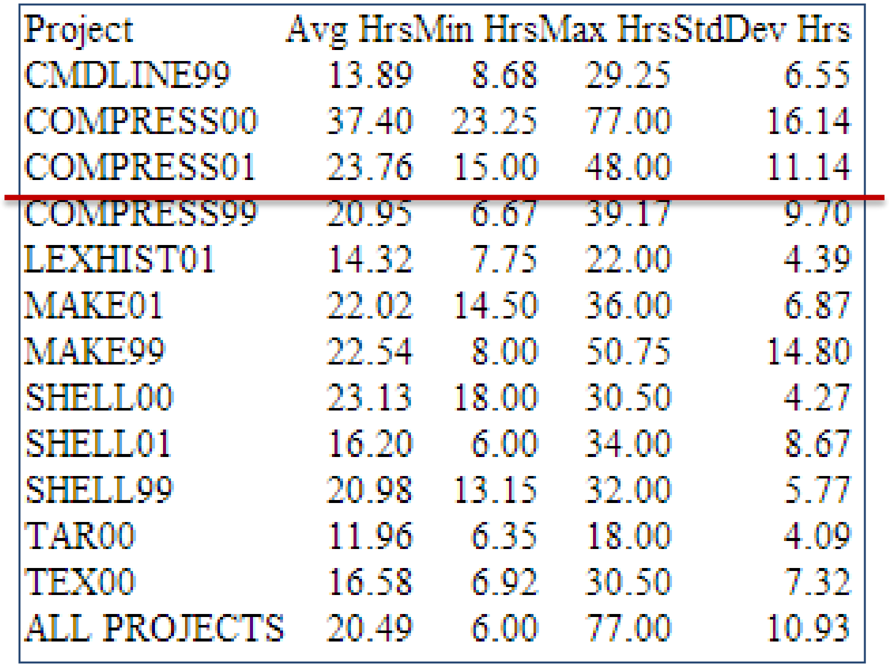
It is not new, however, it is well known in teaching

We Help the World's Best Developers Make Better Software.

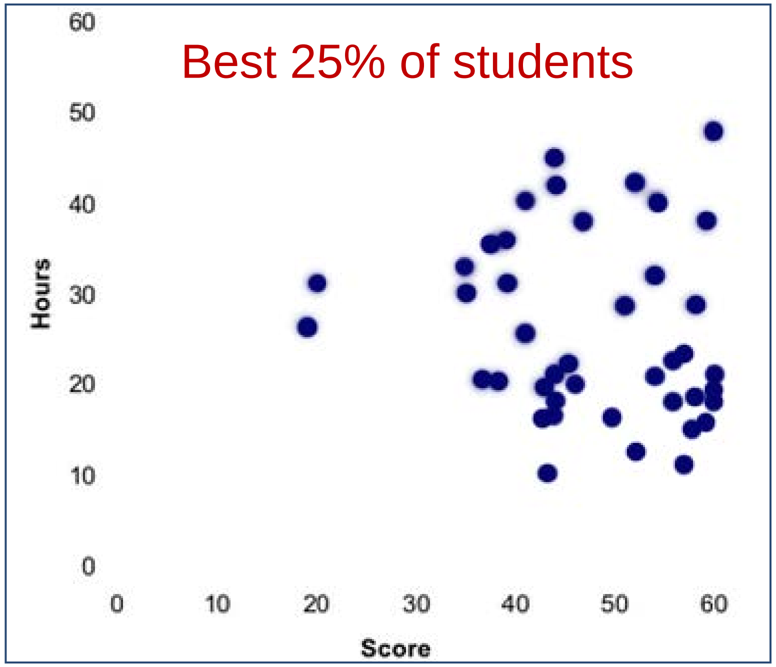
We had a different idea. What if programmers were treated like rock stars? What if management's number one responsibility was recruiting extremely talented software people, treating them well, and then getting the heck out of the way while they did great work?

Read more about our philosophy »

$$\begin{tikzcd}
\boxed{\text{Best Working Conditions}} \arrow[r] & \boxed{\text{Best Programmers}} \arrow[r] & \boxed{\text{Best Software}} \arrow[r] & \boxed{\text{Profit!}}
\end{tikzcd}$$

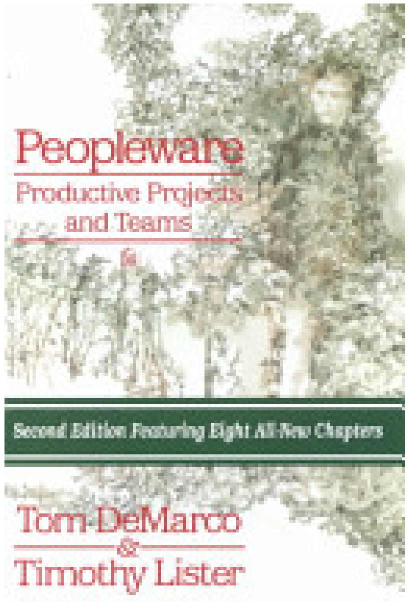


Project	Avg Hrs	Min Hrs	Max Hrs	StdDev Hrs
CMDLINE99	13.89	8.68	29.25	6.55
COMPRESS00	37.40	23.25	77.00	16.14
COMPRESS01	23.76	15.00	48.00	11.14
COMPRESS99	20.95	6.67	39.17	9.70
LEXHIST01	14.32	7.75	22.00	4.39
MAKE01	22.02	14.50	36.00	6.87
MAKE99	22.54	8.00	50.75	14.80
SHELL00	23.13	18.00	30.50	4.27
SHELL01	16.20	6.00	34.00	8.67
SHELL99	20.98	13.15	32.00	5.77
TAR00	11.96	6.35	18.00	4.09
TEX00	16.58	6.92	30.50	7.32
ALL PROJECTS	20.49	6.00	77.00	10.93



http://www.joelonsoftware.com/articles/HighNotes.html

It is also known among practitioners, and they still argue


Peopleware
Productive Projects and Teams
Second Edition Featuring Eight All-New Chapters
Tom DeMarco & Timothy Lister

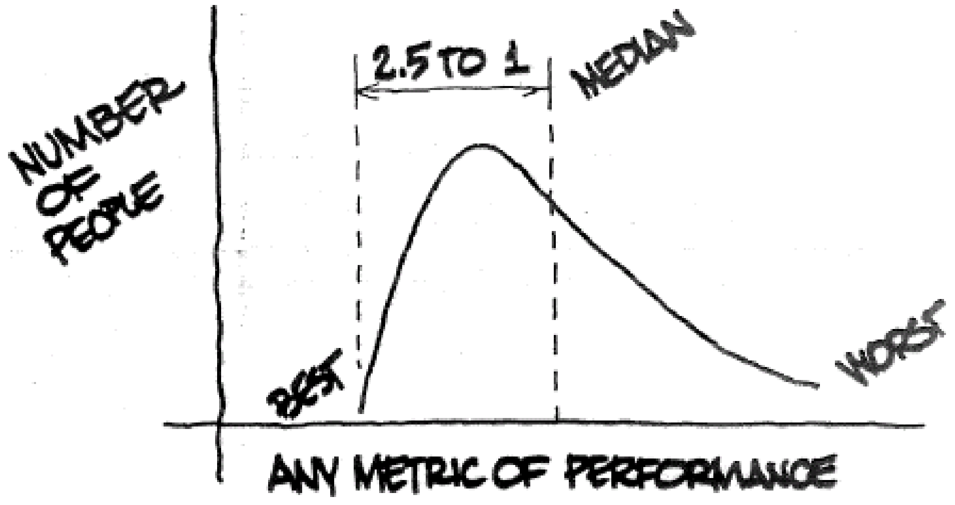
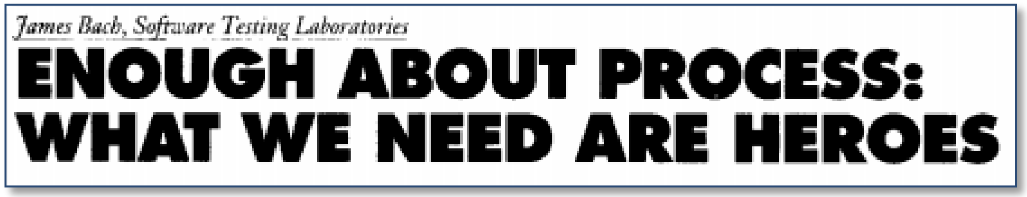
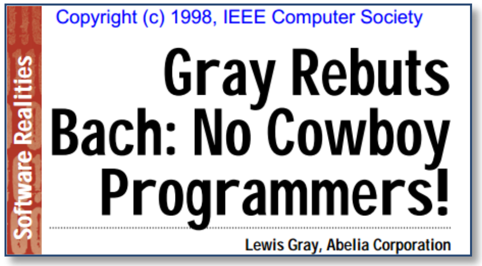


Figure 8.1. Productivity variation among individuals.

"There are order-of-magnitude differences among programmers" has been confirmed by many other studies of professional programmers (Curtis 1981, Mills 1983, DeMarco and Lister 1985, Curtis et al. 1986, Card 1987, Boehm and Papaccio 1988, Valett and McGarry 1989, Boehm et al 2000).


James Bach, Software Testing Laboratories

ENOUGH ABOUT PROCESS: WHAT WE NEED ARE HEROES


Copyright (c) 1998, IEEE Computer Society

Software Realities

Gray Rebuts Bach: No Cowboy Programmers!

Lewis Gray, Abelia Corporation

Software Trajectory Analysis (STA)

- So, we know, that programming is a human driven activity.
- And we know that we are "products" of our habits (recurrent behaviors), we even say, that "Habit is Second Nature".
  - We learn them by heart (from kindergarten), acquire them from others through training, we foster good ones and fight bad ones through life.
- Also, we know, that some programmers are the way better than others, but we don't know why. I argue that their behaviors is that what matters the most for performance.
- What if we would have a powerful mechanism in place which aids the detection of recurrent behaviors, so we could advance our knowledge? And become better.

How to study software processes? (a.k.a. behaviors)

- One way is to invent the processes and conduct experiments in controlled environment
  - extremely expensive, needs massive resources, “spare” teams
    - Need to motivate people to use “paper lions” – invented processes
  - takes decades to finish the full SDLC study

OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!
S. Adams E-mail: SCOTTADAMS@AOL.COM
11/13 © 1995 United Feature Syndicate, Inc.(NYC)

http://universaluclick.com/licensing_permissions/educational_use

How to study software processes?

- Another way is to observe "good" and "bad" programmers and teams drawing conclusions
  - it is an intrusive method which brings a lot of external pressure on people
  - difficult to make unbiased judgment

I WORKED AROUND THE CLOCK AND FINISHED A PROJECT THAT WOULD NORMALLY REQUIRE TEN PROGRAMMERS.
UM... DID I JUST ESTABLISH A NEW BASELINE EXPECTATION THAT WILL TURN MY JOB INTO A TRAGIC DEATH MARCH?
IT'S TIME TO SET SOME STRETCH GOALS.
STUPID! STUPID! STUPID!
www.dilbert.com scottadams@aol.com
9-12-08 ©2008 Scott Adams, Inc./Dist. by UFS, Inc.

http://universaluclick.com/licensing_permissions/educational_use

How to study software processes?

- Yet another way, is to study them offline – by analysis of software process artifacts.
  - It is inexpensive
  - Projects could be studied anytime within SDLC, even post-mortem

- However there are problems:
  - Availability of artifacts, are there any artifacts reflecting behaviors?
  - Granularity of artifacts, do the available artifacts fine enough to reconstruct behaviors?
  - Informational content of artifacts, is there enough information to properly assess generative behaviors?
  - Well, we can study all this: my research hypothesis states that it is possible to infer recurrent behaviors from software process artifacts.

Why is it possible?

- It use to be that many things were kept on paper.
- Space-time constraints use to limit our access to network and computers, we had to travel for meetings; codebase and processes were not public at all. (builds, tests, issues, changes, etc.)

- All changed now. Distance is dead. There is no distance in Cyberspace, we can code, chat, compute - anytime and anywhere, all at the same time - my smartphone does it all.
Code and all the documents can be easily public. (SCM, CI, WiKi, QA sites)

- Along this phenomena, another one arose – collaboration. It drives Linux - #1 computing platform. It drives Android - #1 mobile platform*, Mozilla, Google Chrome, MySQL, Postgre, R, Octave … you name it
  - **Source code is public. Documents are public. Discussions/Issues, etc.**
  - **Commit often, Release early, Use it, Be responsive to users…**

*http://edition.cnn.com/2012/12/13/tech/mobile/google-schmidt-android-ios

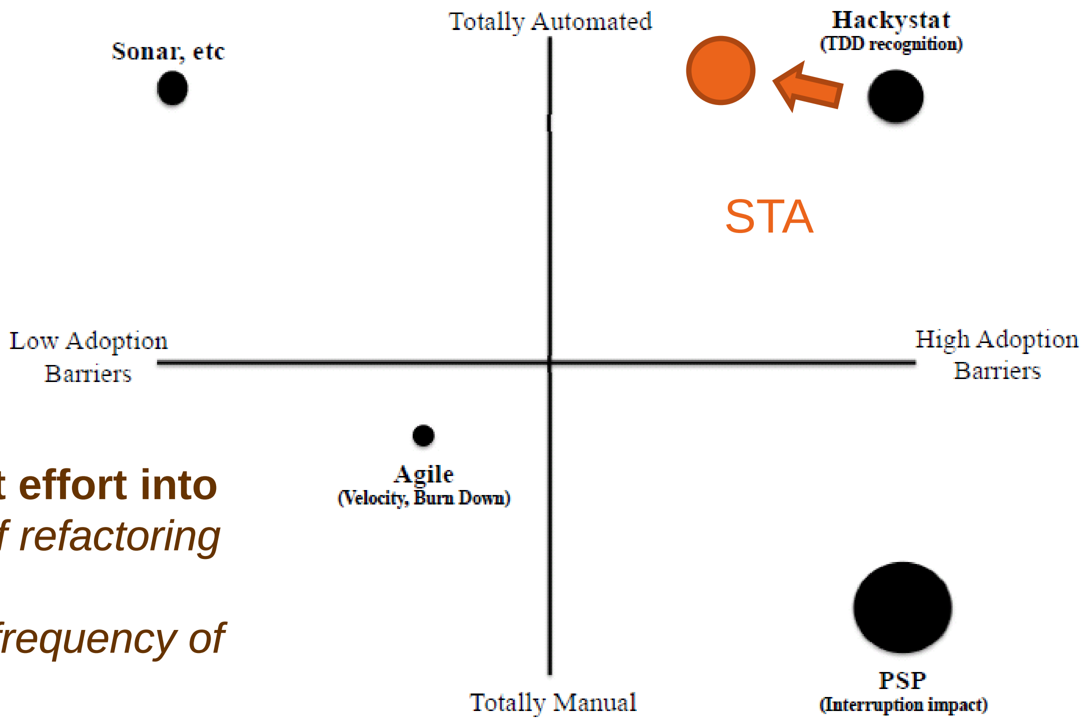
STA Approach: behaviors dictionary

- Fortunately, with the use of software change management and with current publicly open repositories, there is a lot of data around to explore the hypothesis
  - Source code repositories
  - Bug/Issue tracking systems
  - Programmers forums, QA sites
- STA takes approach of building programmers "activity dictionaries" following natural time intervals and using software metrics
  - Daily intervals (also nights, mornings, working day, evening, late night)
  - Weekly intervals
  - Monthly intervals
  - SDLC cycle intervals (release, migration, update)
- Once dictionaries built and indexed, KDD tools applied in order to find significant activities.
  - Recurrent activities (time-series motifs)
  - Individual (or group) specific recurrent activities (time-series motifs)
  - Rare behaviors (time-series discords)

Hey! We seen that before!

- Yes, it is similar to Hackystat, **in fact - it is built for Hackystat**, and yes, STA wants to quantify one's **effort** (**very personal data**).
- No, it is different from Sonar, Moose, Ohloh, DevCreek etc. I am not particularly interested in product quality, rather in **metrics as effort derivative**.

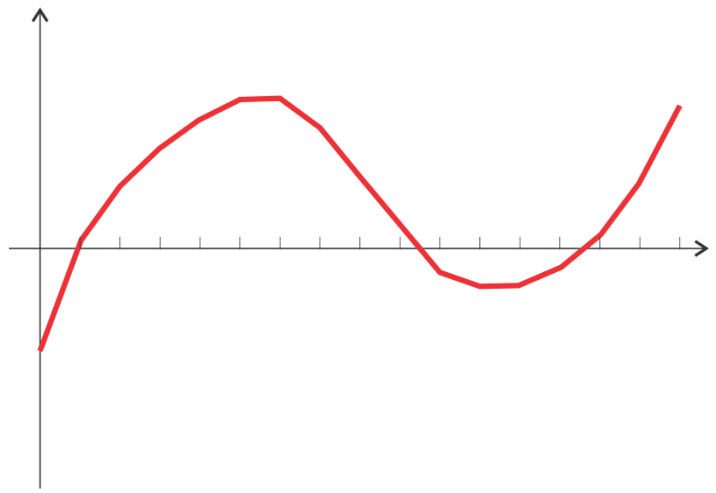
Figure 6: A three dimensional classification for software analytics approaches, including automation, adoption barriers, and breadth of possible analytics supported by the approach (indicated by the size of the circle).



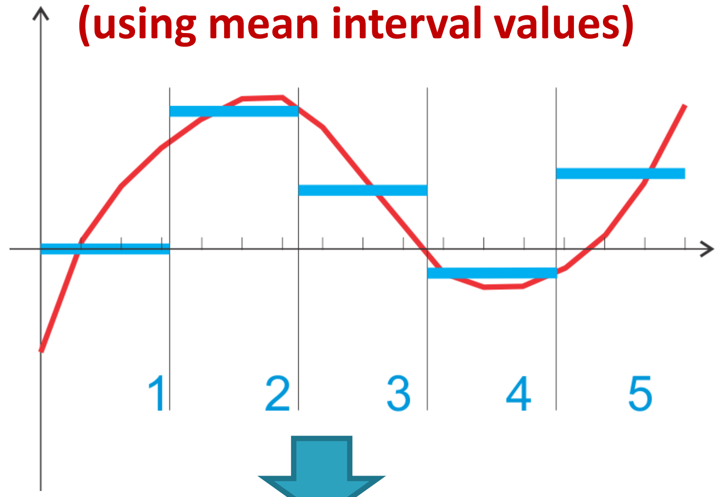
Maybe one would say:
« …**Hey, when we put consistent effort into API refactoring** (*high frequency of refactoring behaviors among many people*) –
our tests seem to fail less (*high frequency of good success/failure behaviors*)

"Looking under the lamppost for useful software analytics" Philip Johnson, https://csdl-techreports.googlecode.com/svn/trunk/techreports/2012/12-11/12-11.pdf

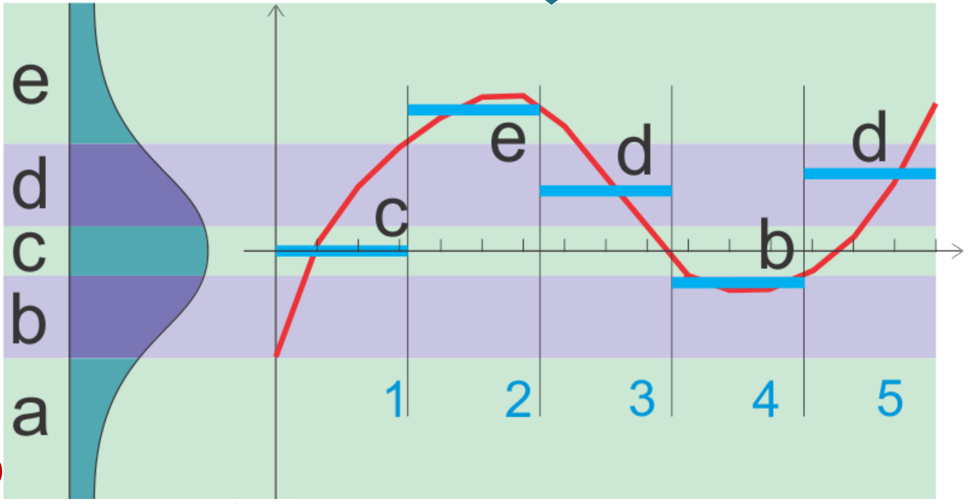
STA pre-processing: Timeseries => strings



Piecewise Aggregate Approximation (using mean interval values)

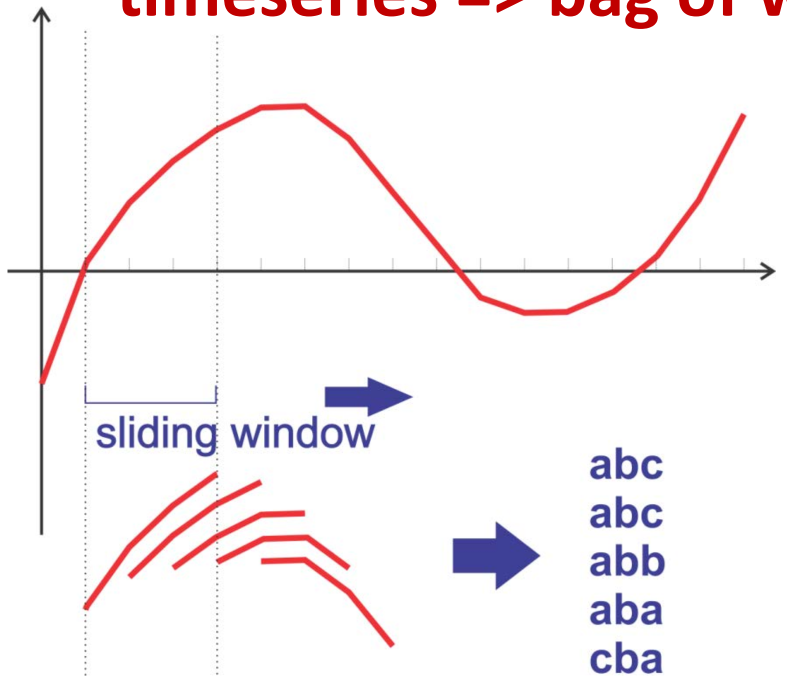


- PAA takes care about lost values averaging extremes
- Strings:
  - easy to handle
  - easy to index
  - Search? RegEx!
  - Edit/Levenshtein distance
    - It is lower bounding to Euclidean
  - So many String algorithms ☺
  - Bioinformatics tools ☺



Symbolic Aggregate Approximation: mean value to letter, using a lookup table

timeseries => bag of words => vector space model



As subseries extracted, they are converted into SAX words. All these words together compose a “bag of words” – quite specific to the series.

Multiple real-valued data streams are converted into “bags of words” – a very convenient representation which allows to “re-use” the wealth of information retrieval research field.

While converting timeseries to bag of words, two tricks are applied:

- the first one is that every sliding window is Z-normalized, i.e. “normalized by energy”. This trick allows STA “focus” only on changes in the event flow, rather than on the amplitude.
- the second is that not all the words are accepted into the bag, if consecutive word is the same as the previous, STA rejects that word from the bag. Thus, for example, continuous growth segment would be represented by only one (first) word.
- Third trick in the ToDo stack is to use sets of window/paa/ alphabet of different sizes when building the same bag – so I can catch variety of “frequencies”

Vector space model, TF*IDF

$D - \textit{document corpus}$

$d - a \textit{ document, "bag" of SAX words}$

$t - a \textit{ term, i.e. a SAX word}$

$$TF(t,d) = \frac{f(t,d)}{\max\{f(w,d) : w \in D\}}$$

$\textit{where } f(t,d) \textit{ is a frequency of the term}$

$$IDF(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

$$TF*IDF(t,d,D) = TF(t,d) * IDF(t,D)$$

$$\begin{tikzcd}
\text{Timeseries} \arrow[d] \\
\text{Bags of SAX words} \arrow[d] \\
\text{tfidf weights vector}
\end{tikzcd}$$

Vector space model, Cosine similarity

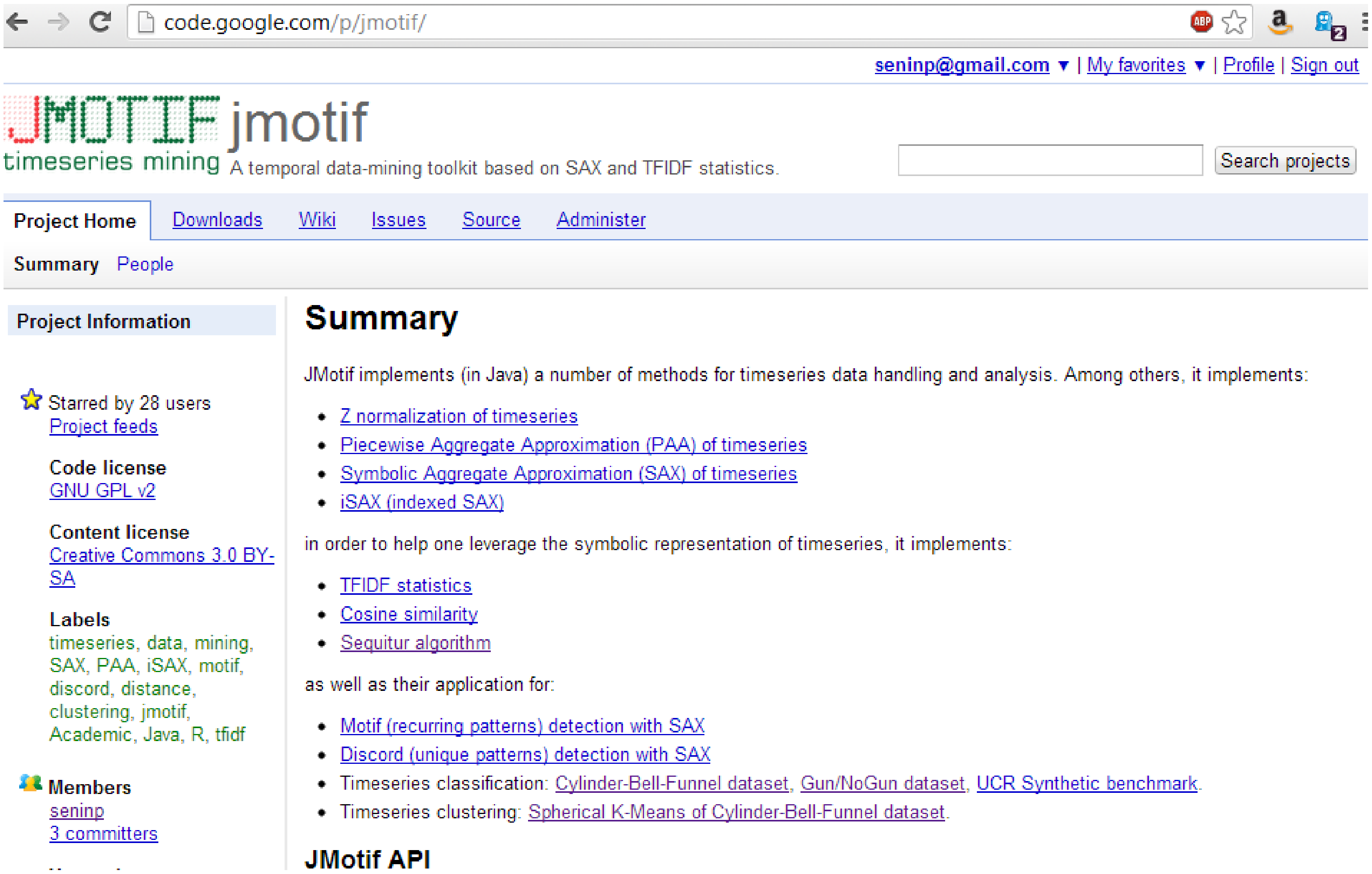
$$similarity = \cos(\theta) = \frac{A \bullet B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} a_i * b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} * \sqrt{\sum_{i=1}^{n} b_i^2}}$$

By using cosine similarity, it is possible to find an angle between two tfidf vectors.

So, it boils down to the fact, that SAX and Vector space model allow us to find an angle between two timeseries – this is the distance.

But how good this metrics is? How does it work, and why it works?

STA relies on Jmotif – a java/R implementation of all above



jmotif

A temporal data-mining toolkit based on SAX and TFIDF statistics.

Project Information

Starred by 28 users
Project feeds

Code license
GNU GPL v2

Content license
Creative Commons 3.0 BY-SA

Labels
timeseries, data, mining, SAX, PAA, iSAX, motif, discord, distance, clustering, jmotif, Academic, Java, R, tfidf

Members
seninp
3 committers

Summary

JMotif implements (in Java) a number of methods for timeseries data handling and analysis. Among others, it implements:

- Z normalization of timeseries
- Piecewise Aggregate Approximation (PAA) of timeseries
- Symbolic Aggregate Approximation (SAX) of timeseries
- iSAX (indexed SAX)

in order to help one leverage the symbolic representation of timeseries, it implements:

- TFIDF statistics
- Cosine similarity
- Sequitur algorithm

as well as their application for:

- Motif (recurring patterns) detection with SAX
- Discord (unique patterns) detection with SAX
- Timeseries classification: Cylinder-Bell-Funnel dataset, Gun/NoGun dataset, UCR Synthetic benchmark.
- Timeseries clustering: Spherical K-Means of Cylinder-Bell-Funnel dataset.

JMotif API

STA performance, CBF domain

$$c(t) = (6+\eta) * \mathrm{X}_{[a,b]}(t) + \varepsilon(t)$$

$$b(t) = (6+\eta) * \mathrm{X}_{[a,b]}(t) * (t-a)/(b-a) + \varepsilon(t)$$

$$f(t) = (6+\eta) * \mathrm{X}_{[a,b]}(t) * (b-t)/(b-a) + \varepsilon(t)$$

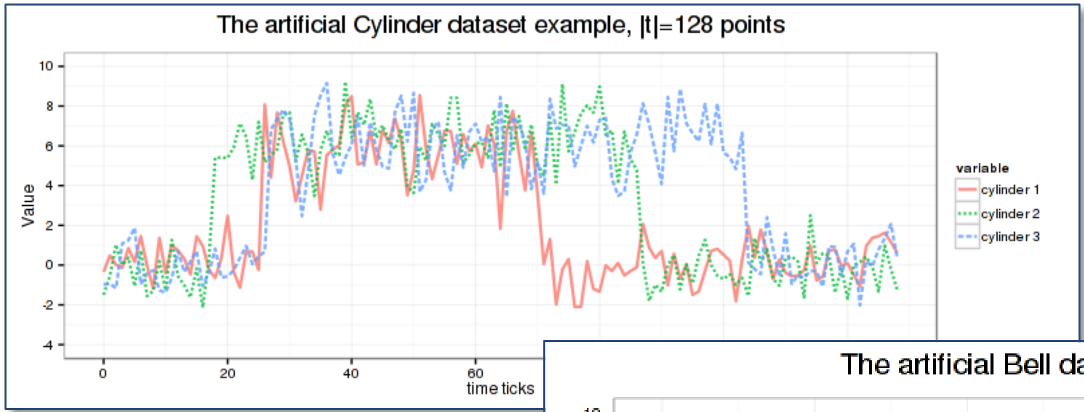
$$\mathrm{X}_{[a,b]} = \begin{cases} 0, t < a \\ 1, a \le t \le b \\ 0, t > b \end{cases}$$

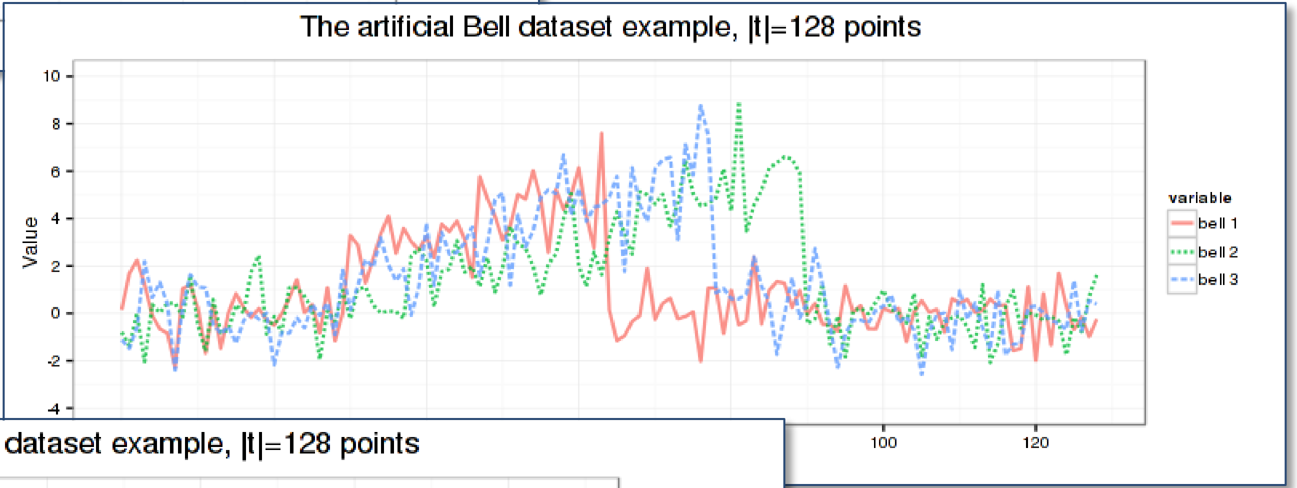
where η and $\varepsilon(t)$ are drawn from $N(0,1)$

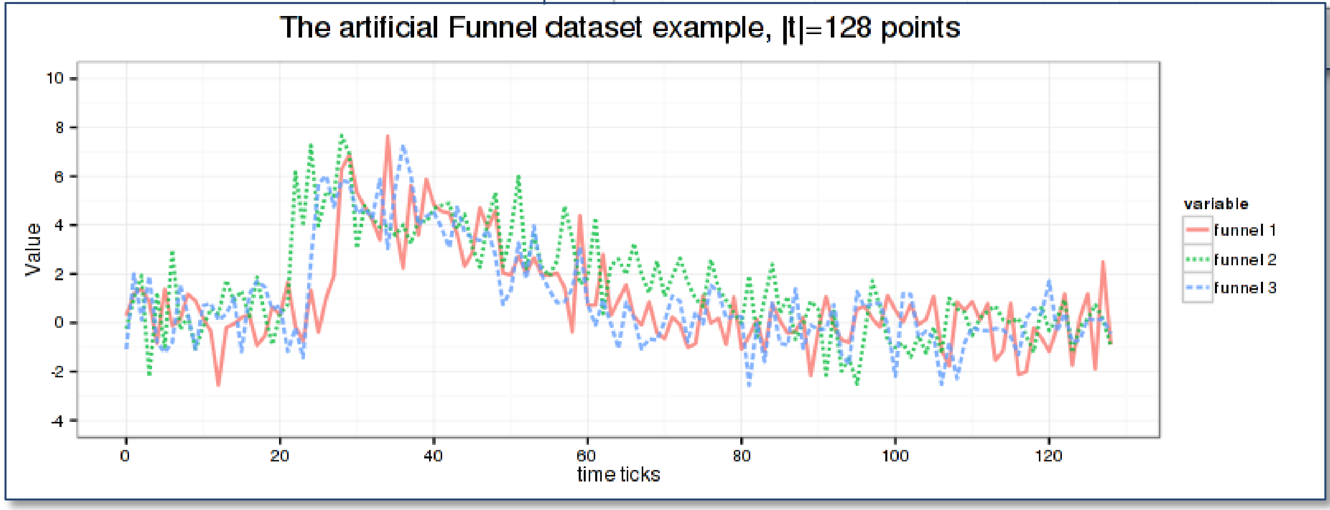
and a is integer uniformly drawn from $[16, 32]$

and $b-a$ is uniformly drawn from $[32, 96]$

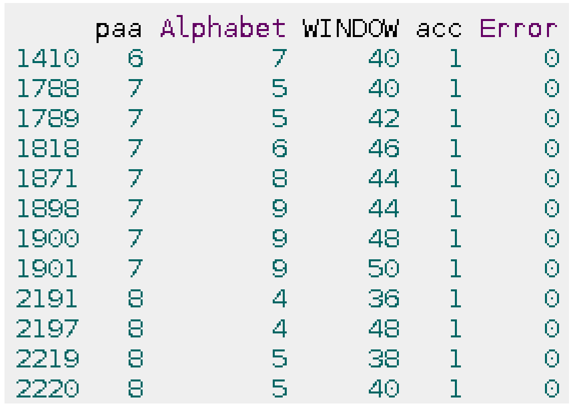
STA performance, CBF domain







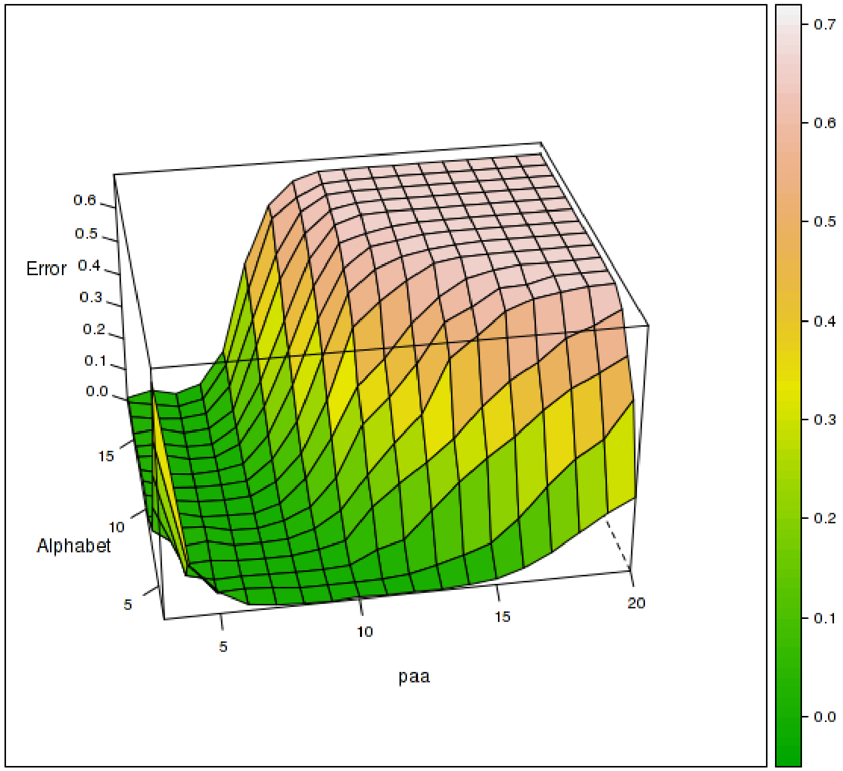
STA performance, CBF domain, Classification



```
     paa Alphabet WINDOW acc Error
1410   6        7     40   1     0
1788   7        5     40   1     0
1789   7        5     42   1     0
1818   7        6     46   1     0
1871   7        8     44   1     0
1898   7        9     44   1     0
1900   7        9     48   1     0
1901   7        9     50   1     0
2191   8        4     36   1     0
2197   8        4     48   1     0
2219   8        5     38   1     0
2220   8        5     40   1     0
```

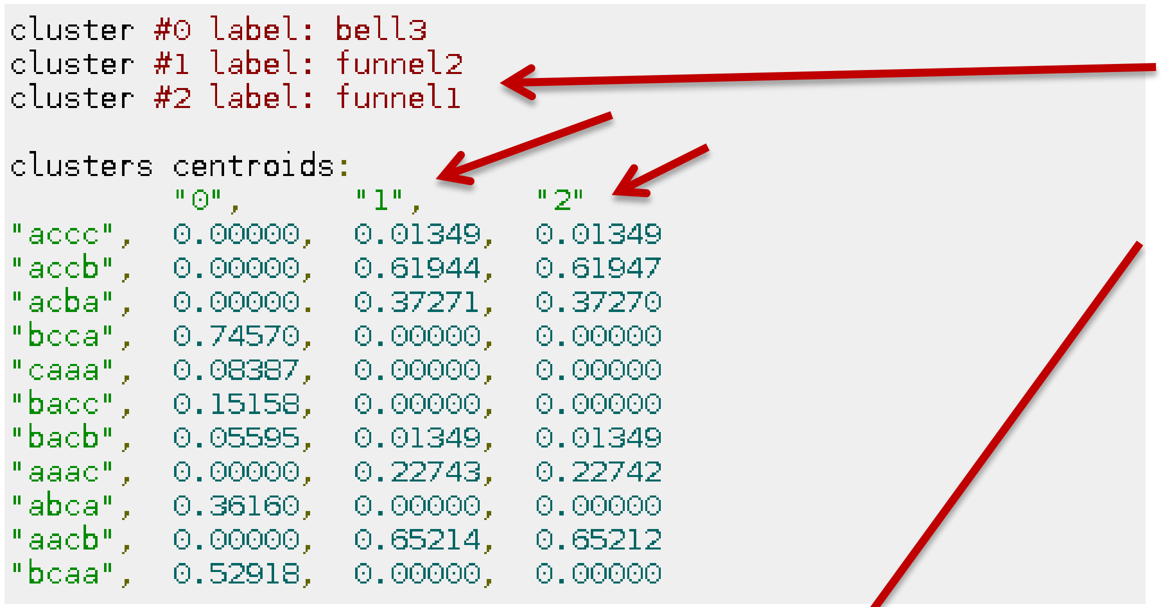
Features/Learner	Published error
Euclidean Distance (Keogh & Kasetty, 2002)	0.003
TClass/Naive Bayes (Kadous, 2002)	0.0367
Segments/Naive Bayes (Kadous, 2002)	0.0620
TClass/C4.5 (Kadous, 2002)	0.019
Segment/C4.5 (Kadous, 2002)	0.0241
TClass with AdaBoost/J48 (Kadous, 2002)	0.0139
Aligned Subsequence (Park et al. 2001)	0.451
Piecewise Normalization (Indyk et al. 2002)	0.130
Autocorrelation Functions (Wang & Wang 2000b)	0.380
Cepstrum (Kalpakis et. al. 2001)	0.570
String (Suffix Tree) (Huang & Yu 1999)	0.206
Important Points (Pratt & Fink 2002)	0.387
Edit Distance (Bozkaya et al.1997)	0.603
String Signature (Jonsson & Badal 1997)	0.444
Cosine Wavelets (Huntala et al. 1999)	0.130
Hölder (Struzik & Siebes)	0.331
Piecewise Probabilistic (Keogh & Smyth 1997)	0.202

CBF Classifier error rate, SLIDING_WINDOW= 44



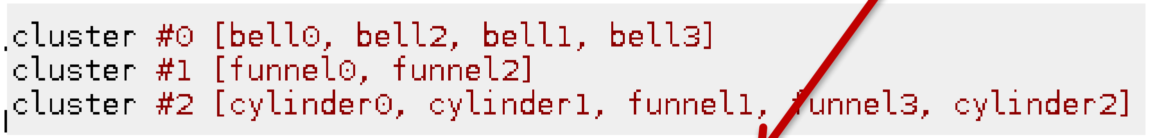
STA performance, CBF domain, Clustering

k-means, updating/normalizing centroids after each iteration "spherical k-means"

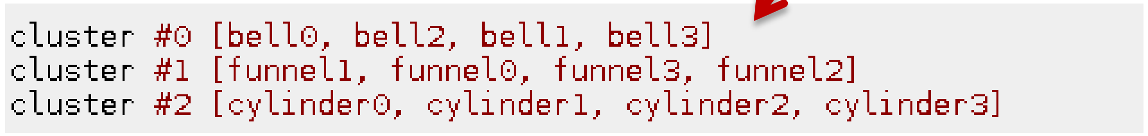


```
cluster #0 label: bell3
cluster #1 label: funnel2
cluster #2 label: funnel1

clusters centroids:
         "0",      "1",      "2"
"accc",  0.00000,  0.01349,  0.01349
"accb",  0.00000,  0.61944,  0.61947
"acba",  0.00000.  0.37271,  0.37270
"bcca",  0.74570,  0.00000,  0.00000
"caaa",  0.08387,  0.00000,  0.00000
"bacc",  0.15158,  0.00000,  0.00000
"bacb",  0.05595,  0.01349,  0.01349
"aaac",  0.00000,  0.22743,  0.22742
"abca",  0.36160,  0.00000,  0.00000
"aacb",  0.00000,  0.65214,  0.65212
"bcaa",  0.52918,  0.00000,  0.00000
```



```
cluster #0 [bell0, bell2, bell1, bell3]
cluster #1 [funnel0, funnel2]
cluster #2 [cylinder0, cylinder1, funnel1, funnel3, cylinder2]
```



```
cluster #0 [bell0, bell2, bell1, bell3]
cluster #1 [funnel1, funnel0, funnel3, funnel2]
cluster #2 [cylinder0, cylinder1, cylinder2, cylinder3]
```

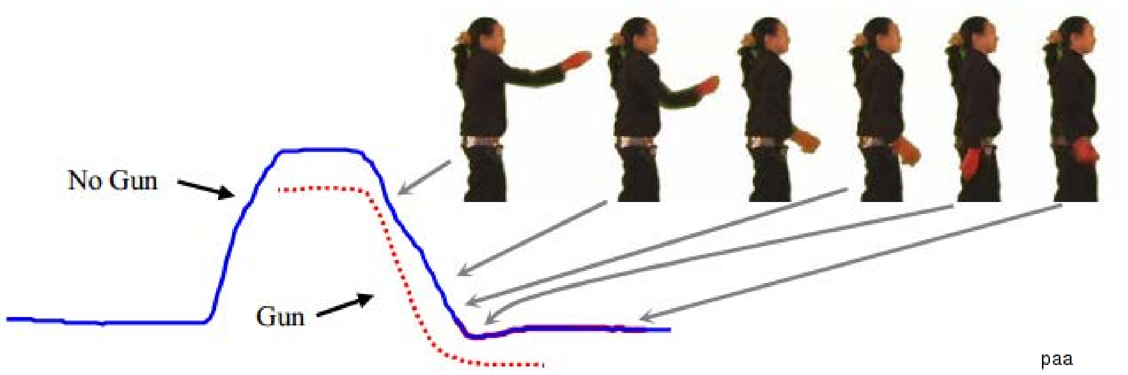
Random centroids assignment made two clusters of the same class

Nevertheless, clustering recovered

Currently I employ further first strategy and restarting
It works better…

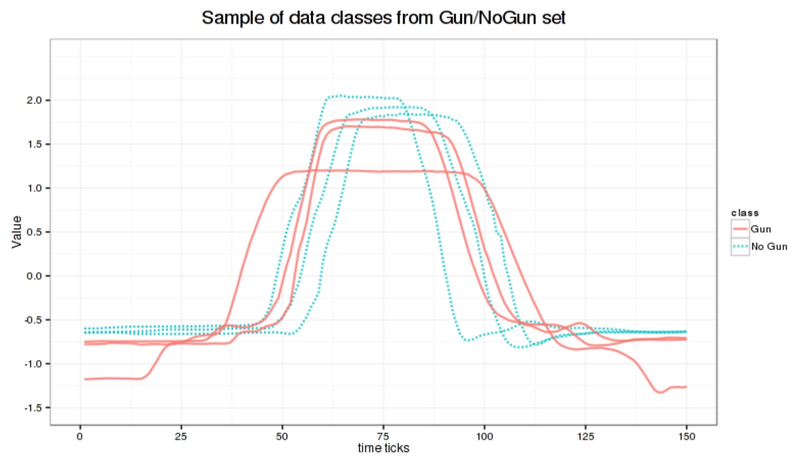
There are online learners to put into, they are well studied and work much better.

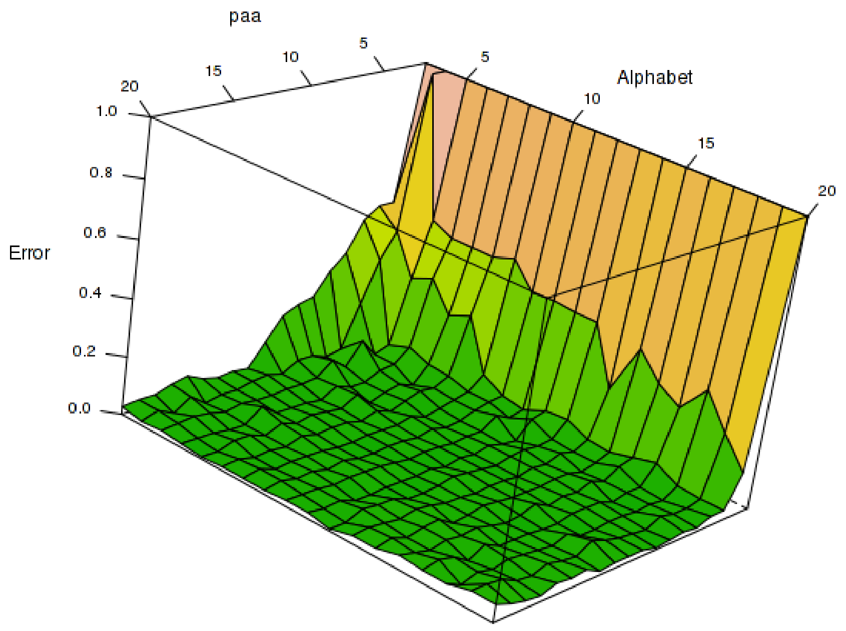
STA performance, CBF domain, Gun/No gun



best classifier:
"1-NN DTW, no Warping Window"
0.7% error rate

STA is the same, while for a number of parameters it is 0%

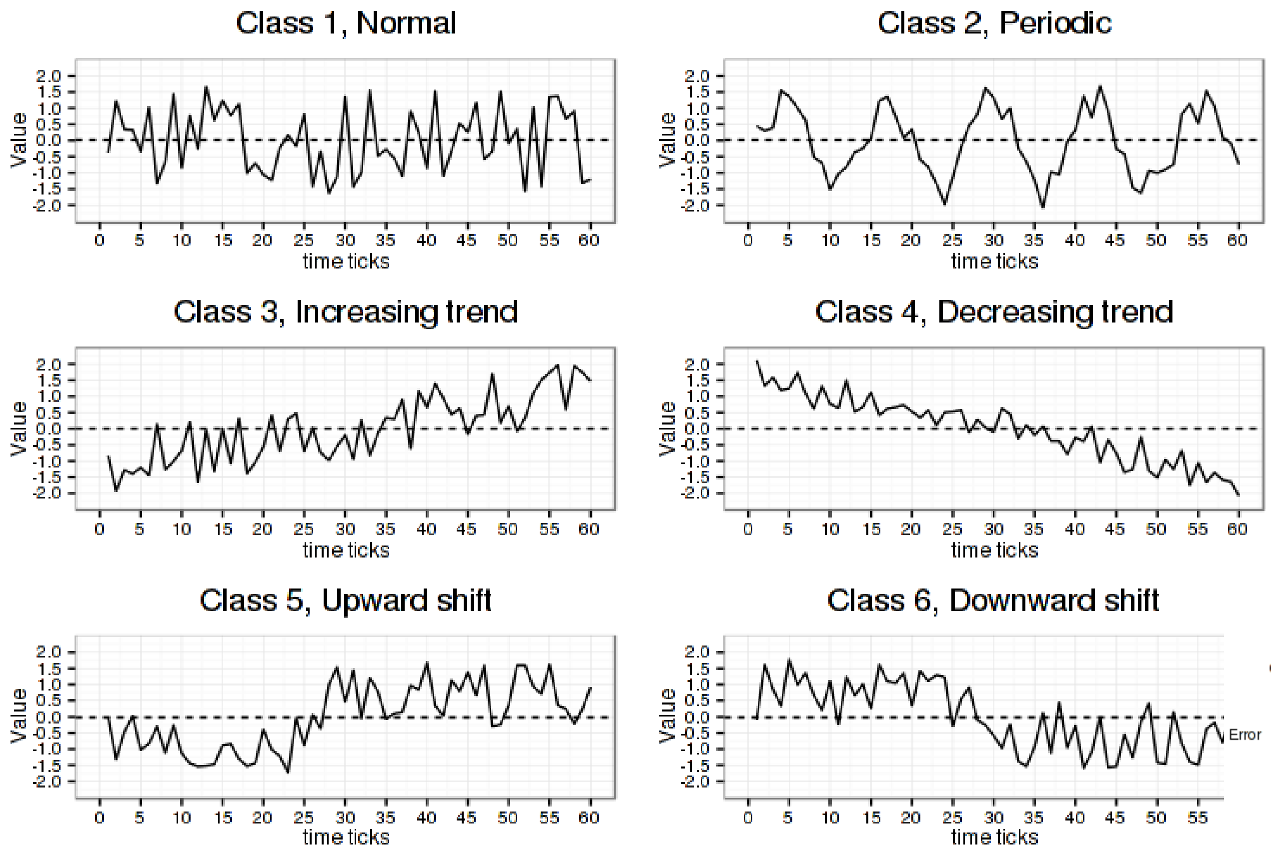


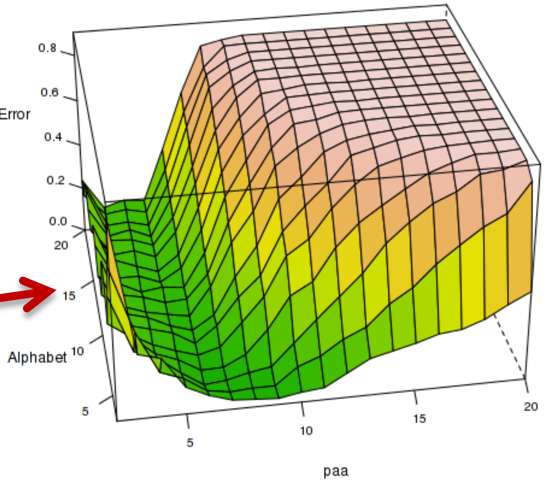


		Knn	NB	C4_5	MLP	RandForest	LMT	SVM
Gun_Point		8,00%	21,33%	22,67%	6,67%	10,67%	20,67%	20,00%

However, STA performance is relatively poor with many classes

but it is designed with different purpose - to simplify indexing and mining





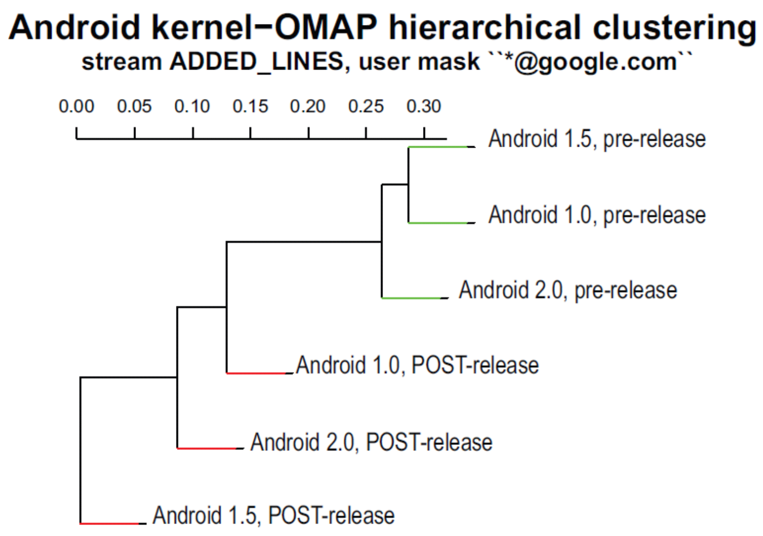
The good parameters valley is very narrow

(but performance is equal to the best classifier)

http://kdd.ics.uci.edu/databases/synthetic_control/synthetic_control.data.html

STA Application: Release cycle in Android

https://csdl-techreports.googlecode.com/svn/trunk/techreports/2012/12-11/12-11.pdf

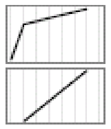
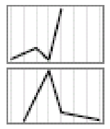
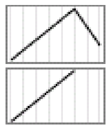
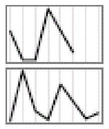
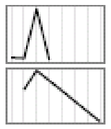
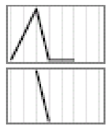
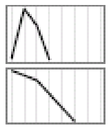
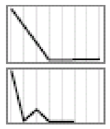
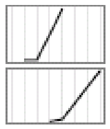
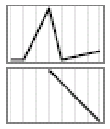
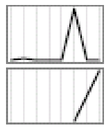

Android kernel−OMAP hierarchical clustering
stream ADDED_LINES, user mask ``*@google.com``
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Android 1.5, pre-release
Android 1.0, pre-release
Android 2.0, pre-release
Android 1.0, POST-release
Android 2.0, POST-release
Android 1.5, POST-release

Release	Classification
1.6-pre	misclassified
1.6-post	OK
2.2-pre	OK
2.2-post	OK
1.1-pre	OK
1.1-post	OK
2.3-pre	OK
2.3-post	OK

Release	Classification
beta -pre	OK
beta-post	OK
2.0.1-pre	OK
2.0.1-post	misclassified
2.1-pre	OK
2.1-post	OK
2.2.1-pre	OK
2.2.1-post	misclassified

Slow weeks and sharp rizes (bug fixes?) in post-release,

«hot» Mondays and busy weekends in pre-release

release	"bbac"	"abca"	"babc"	"bbba"	"bcaa"	"bcbb"	"ccaa"	"cbaa"	"bbcb"	"bbbb"	"bbbc"
post-2.0	0.63	0	0.63	0	0	0	0	0.39	0.24	0.06	0
post-1.0	0	0.93	0	0	0	0	0	0	0	0.09	0.36
post-1.5	0	0	0	0	0	0	0	0	0.79	0.61	0
pre-1.5	0	0	0	0.23	0.23	0.91	0	0.14	0.18	0	0.09
pre-2.0	0	0	0	0	0	0	0	0	0	1	0
pre-1.0	0	0	0	0	0	0	0.79	0	0	0.08	0.61
unnormalized sample curves corre-sponding to patterns											

John Skeet phenomena

meta stackoverflow

Jon Skeet Facts

- Jon Skeet IS the traveling salesman. Only he knows the shortest route
- When Jon Skeet points to null, null quakes in fear
- When Jon Skeet's code fails to compile the compiler apologizes
- When Jon gives a method an argument, the method loses
- When invoking one of Jon's callbacks, the runtime adds "please"
- Drivers think twice before they dare interrupt Jon's code

So, what do I need to do to get in proximity of #1 user of StackOverflow?

"…Often two answers may look quite similar, but one just about has an edge on the other - either it's explained just that bit better, or has one more piece of information, or a code sample. I'd like to *hope* that I have that sort of edge, and that that's why my answer would get more votes in that situation. But hey, I could easily be wrong…"

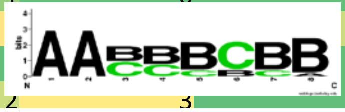
This is right, but how much time/effort it will cost to me?

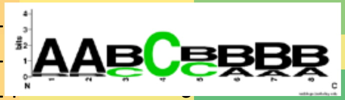
Daily Answers counts for four StackOverflow Rock-stars

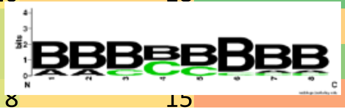
Sliding window=24hrs

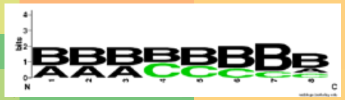
PAA=8 (*bin=3 hrs*), Alphabet=3

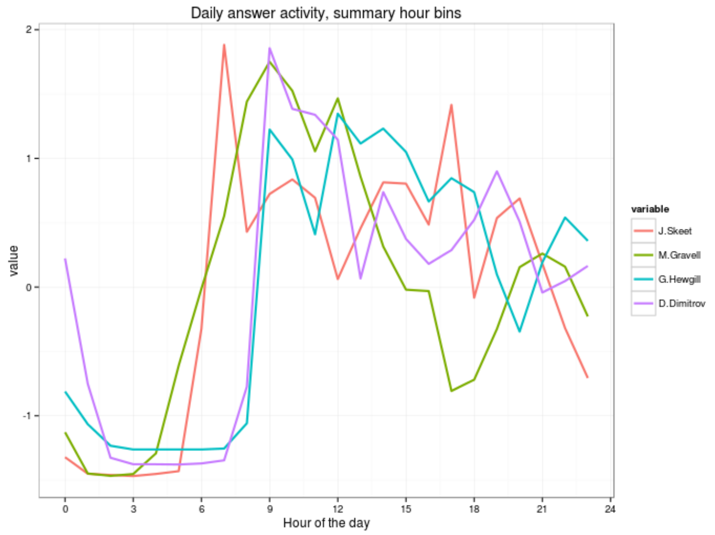
	J.Skeet	M.Gravell	G.Hewgill	D.Dimitrov
1 aabbbccb	36	3	0	0
2 aaccbbbb	30	13	3	0
3 aabbccbb	30	2	5	0
4 aabccbbb	29	19	6	6
5 aacbbcbb	29	0	3	0
6 aabbbcbb	27	5	7	0
7 aaccbcbb	24	1	0	0
8 aabcbcbb	23	[illegible]	[illegible]	2
9 aacbbccb	23	[illegible]	[illegible]	0
10 aacbbcba	22	2	3	0
11 aabcbbbb	18	27	19	1
12 bbbcbbbb	1	27	52	29
13 aabccbbb	29	19	6	6
14 aaccbabb	8	16	3	0
15 aaccbbba	21	15	5	0
16 aabccabb	5	15	2	0
17 aabccbab	1	[illegible]	[illegible]	2
18 aabccbba	20	[illegible]	[illegible]	6
19 aacccbaa	6	[illegible]	[illegible]	0
20 aabcbabb	3	14	3	0
21 bbcbbbbb	4	9	57	1
22 bbbcbbbb	1	27	52	29
23 bbbbcbbb	0	6	26	17
24 bbbccbbb	1	9	24	14
25 aabcbbbb	18	27	19	1
26 bbbbbbcb	4	10	18	53
27 bbccbbbb	2	10	18	1
28 bbbbbcbb	2	[illegible]	[illegible]	20
29 aabccbaa	3	[illegible]	[illegible]	3
30 bbbbbbbc	3	8	15	35
31 bbbbbbcb	4	10	18	53
32 bbbbbbbc	3	8	15	35
33 aaaccbbb	5	9	3	30
34 bbbcbbbb	1	27	52	29
35 bbbbbcbb	2	7	18	20
36 aaaccbba	0	5	3	20
37 bbbbcbbb	0	[illegible]	[illegible]	17
38 aaacccba	0	[illegible]	[illegible]	17
39 aaacbcbb	4	[illegible]	[illegible]	16
40 bbbbbbcc	3	3	12	15











	J.Skeet	M.Gravell	D.Dimitrov	G.Hewgill
J.Skeet	0	0	0	0
M.Gravell	0,1699	0	0	0
D.Dimitrov	0,0096	0,04568	0	0
G.Hewgill	0,13533	0,18094	0,1533	0

STA tells us, that Skeet is almost orthogonal to other folks, and… he sleeps at night!!!
But:

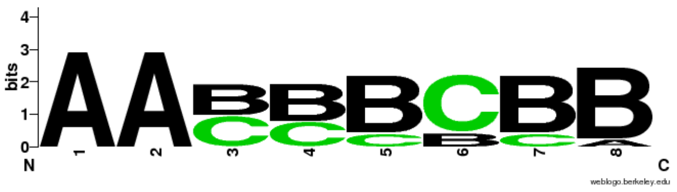
- **5 bins of ''C'' s**
- **peaks before and after working day**

http://weblogo.berkeley.edu/logo.cgi

John Skeet phenomena

Five C bins and peaks before and after working day:

- Stays active throughout a day
- Answers a lot before/after work

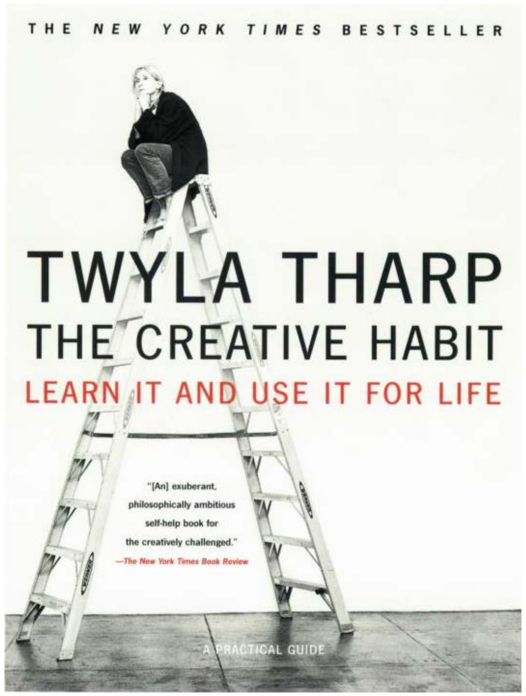


"…I have a longish commute both ways each day: a 3G data dongle lets me answer questions during that time. I spend a fair amount of time in the evening on my computer for whatever reason (coding, writing talks or articles, etc) - I pop onto SO every so often.
(this is a green C on top row!)
While at work, I tend to check SO while I have tests running, a deploy, or a build. I hope my colleagues wouldn't regard me as a slacker though…" (five green Cs)

So, want to be a number one? – take a train to work, and continuously work through evenings, plus – keep an eye on SrtackOverflow during your day…

But also, "my answers … explained just that bit better, or has one more piece of information, or a code sample…"

Well, this is what we already seen – be passionate about what you do + develop a habit, and keep it for life. (take away points)


THE NEW YORK TIMES BESTSELLER
TWYLA THARP
THE CREATIVE HABIT
LEARN IT AND USE IT FOR LIFE
"[An] exuberant, philosophically ambitious self-help book for the creatively challenged."
—The New York Times Book Review
A PRACTICAL GUIDE

Importance of good habit:

…Turning something into a ritual eliminates the question:

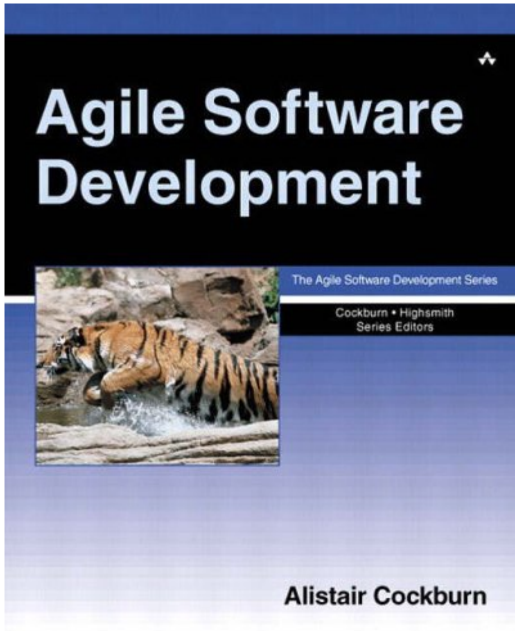
Why am I doing this?

"…by the time I give a taxi driver directions, it's too late to wonder why I am going to the gym and not snoozing under warmth of my bed. The cab is moving. I am committed. Like it or not, I am going to gym…"

View on methodologies:

"…All methodology is based on fears…"

"…A methodology's principles are not arrived at through an emotionally neutral algorithm but come from the author's personal background…
One can almost guess at a methodology author's past experiences by looking at the methodology…."


Agile Software Development
The Agile Software Development Series
Cockburn • Highsmith Series Editors
Alistair Cockburn

STA contribution and future directions

- Novel methodology for recurrent behaviors discovery

- Novel timeseries mining method and its implementation
    - Well, technically speaking, it was known before, there are published research work _mentioning_ its application.
      I seen at least two papers.
      «A Dimensionality Reduction Technique for Efficient Time Series Similarity Analysis»
      http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2390719/

- Case studies partially confirming the research hypothesis – it is possible to recover recurrent behaviors from software process artifacts.

- Methodology and the algorithm could be extanded further – multidimentional series converted into bags of words easily.
    - Letter for the prefix, «Z» - test coverage, «Y» - DevTime

- More case studies, maybe industrial studies, classroom experiments.

Thank you!

My adviser

Philip Johnson
Professor (ICS)
Associate Department Chair
Specialty: Software Engineering, Renewable Energy
Office: POST 307A
Phone: 808-956-3489
E-mail: johnson@hawaii.edu

And my department

University of Hawaii at Manoa Information & Computer Sciences

Backup slides, Additional info data, and all the stuff which I think could be useful

Programmer, Developer, Engineer

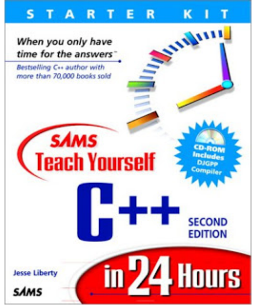
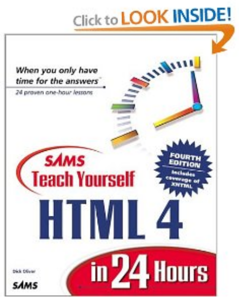
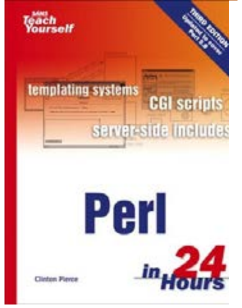
- **programmer** = we do not have many technical employees and need someone to "program" something; for example a law firm*, i.e. "a person who knows how to write code*

- **developer** = we are a tech-savvy product or services company and need someone to work on internal or back-end tools; for example a bank or consulting company

- **engineer** = we are a software company and need someone to work on one of our products; for example Adobe or Microsoft… i.e. *"a person who has studied software engineering or computer science“, (usually +$20K to programmer’s average)*
- Among all the opinions and positions, majority agrees (I am citing StackOverflow again) that the title is mostly defined to **one’s ability** not only being able to write the code, but to be comfortable with a large codebase and to deliver and support a product.

http://stackoverflow.com/questions/27516/whats-the-difference-between-programmer-and-software-engineer

Recent shift in education:

Teach Yourself Programming in Ten Years by Peter Norvig…

…Researchers (Bloom (1985), Bryan & Harter (1899), Hayes (1989), Simmon & Chase (1973), have shown it takes about ten years to develop expertise in any of a wide variety of areas, including chess playing, music composition, telegraph operation, painting, piano playing, swimming, tennis, and research in neuropsychology and topology...

Programming:

- Get interested.
- Program. The best kind of learning is learning by doing.
- Talk with other programmers; read other programs.
- If you want, put in four years at a college.
- Work on projects with other programmers.
- Work on projects after other programmers.
- Learn at least a half dozen programming languages....

Other known cost overruns

- Ballistic Missile Early Warning System, moonrise caused enough worries to modify system
- Mars Climate Orbiter 23 September 1999 Orbiter crash landed on surface due to metric-imperial mix-up
- Great Britain, the NHS National Programme for IT ~30 billion USD
- Canadian Firearms Registry
  - in 2004 overall program cost: ~2B, fees collected: ~140M
  - estimated running cost $2 million per year, ~$70M in 2011
- Finnish Vehicle Administration ~70 million USD overrun, 8 years late
- Bank of America, MasterNet: 23M for system development, 600M make it run = cancelled, lost customers and profit?
- Netscape 6
- Duke Nukem Forever