Analogy Based Prediction of Work Item Flow in Software Projects: a Case Study

Audris Mockus

audris@avaya.com

Avaya Labs Research

Basking Ridge, NJ 07920

http://www.research.avayalabs.com/user/audris

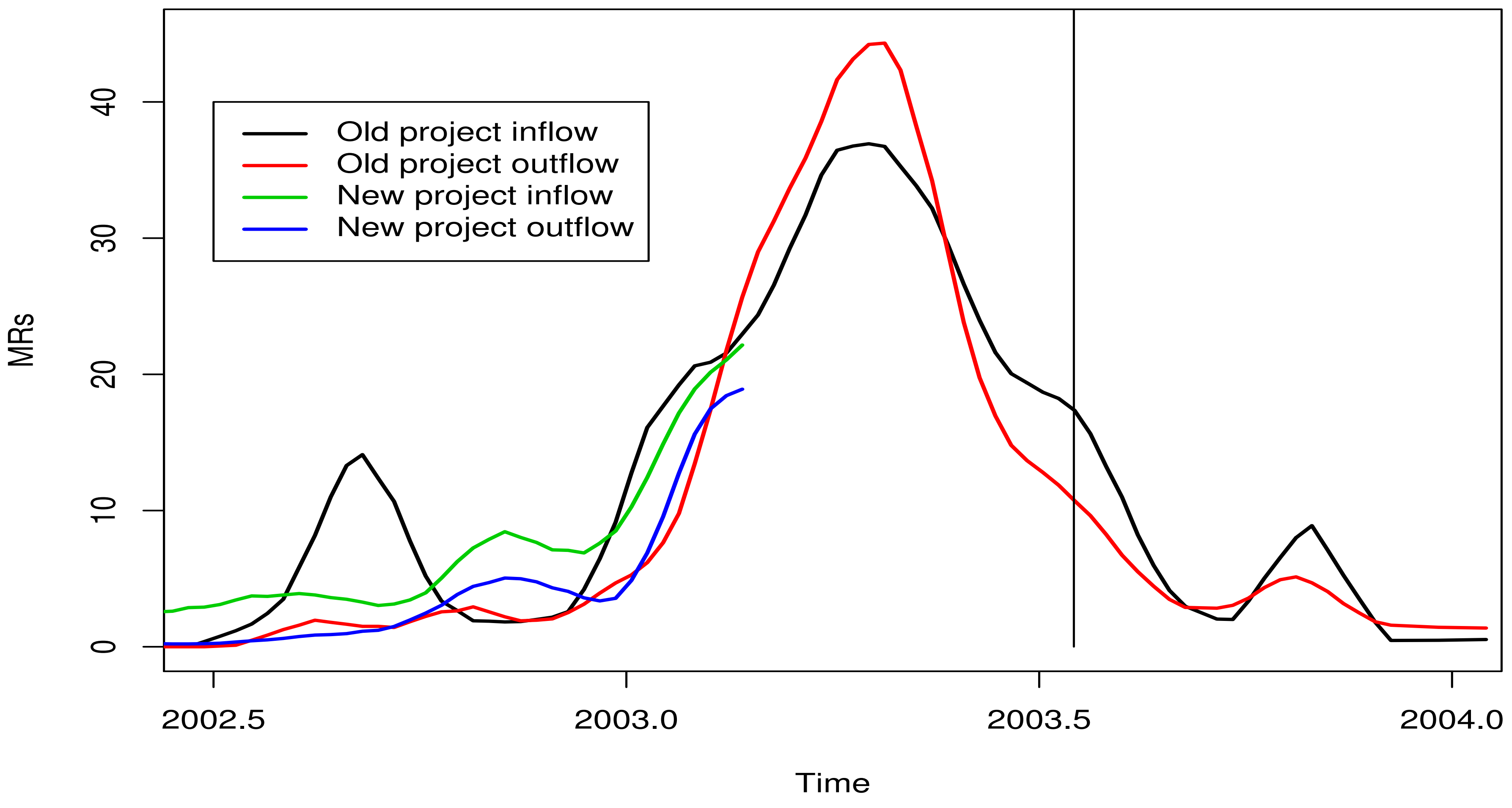
Notes

- This is an experiment to report ongoing work in a conference setting (specially designed for ISESE) as a contrast for ICSE and most other fora
  - The work started about 9 months ago
  - First results (prediction system/predictions) were obtained 6 months ago (ISESE submission)
  - Revised 2.5 months ago (see in proceedings)
  - The latest results are exactly one week old
  - The latest slides are no more than 1 hour old
- Let’s try to be interactive
- Apologies for messy slides/incoherent presentation
- The talk is not about effort estimation

Motivation

- To quantify software production: make informed trade-offs between schedule, quality, cost
  - Visibility: where/when effort is spent, defects introduced
  - Predictability: what will be the impact of choosing technology, processes, organization
  - Controllability: trade-offs between time to market, features, quality, and staffing
- Desire to explain observed software phenomena that are increasingly represented by logs of workflow/knowledge management systems (earlier we heard about Ticho Brage and astronomical observations)

Predicting Defect Counts



Background: Illustration

- Software is created by work items or changes
- Changes are tracked to enable multiple people to work on them
- The workflow represents various relationships

Why Use Project Repositories?

- The data collection is non-intrusive (using only existing data minimizes overhead)
- Long history of past projects enables historic comparisons, calibration, and immediate diagnosis in emergency situations.
- The information is fine grained: at MR/delta level
- The information is complete: everything under version control is recorded
- The data are uniform over time
- Even small projects generate large volumes of changes: small effects are detectable.
- The version control system is used as a standard part of a project, so the development project is unaffected by observer

Pitfalls of Using Project Repositories

- Different process: how work is broken down into work items may vary across projects
- Different tools: CVS, ClearCase, SCCS, ...
- Different ways of using the same tool: under what circumstances the change is submitted, when the MR is created
- The main challenge: create change based models of key problems in software engineering

Existing Models

- Predicting the quality of a patch [6]
- Globalization: move development where the resources are:
  - What parts of the code can be independently maintained [7]
  - Who are the experts to contact about any section of the code [5]
- Effort: estimate MR effort and benchmark process
  - What makes some changes hard [3]
  - What processes/tools work [1, 2]
  - What are OSS/Commercial process differences [4]
- Project models
  - Release schedule [8]
  - Release readiness criteria
  - Release quality

Change Data Methodology: Project Sample

- ✦ *Languages*: Java, C, SDL, C++, JavaScript, XML, ... *Platforms*: proprietary, unix'es, Windows, VXWorks, *Domains*: embedded, high-availability, network, user interface *Size*: from largest to small

Type	Added KLines	KDelta	Years	Developers	Locations
Voice switching software	140,000	3,000	19	6,000	5
Enterprise voice switching	14,000	500	12	500	3
Multimedia call center	8,000	230	7	400	3
Wireless call processing	7,000	160	5	180	3
Web browser	6,000	300	3	100/400	
OA&M system	6,000	100	5	350	3
Wireless call processing	5,000	140	3	340	5
Enterprise voice messaging	3,000	87	10	170	3
Enterprise call center	1,500	60	12	130	2
Optical network element	1,000	20	2	90	1
IP phone with WML browser	800	6	3	40	1
Web sever	200	15	3	15/300	

Objective

- Predict workflow in a software project
  - To obtain the number of MRs that will arrive in various queues before and after the release date
  - To obtain the number of MRs remaining in various queues at release date
  - To obtain schedule of work inflow/outflow including the release date
- Context
  - The project uses its own CRM product to track workflow including functional requirements, software defects/features, and customer calls, patches
  - Project completion/readiness/quality criteria are based on workflow summaries
  - 10 year, 35K MRs, 5M lines of $C++$, Java, C, 80 developers, 60 testers/mngmnt/field support

Assumptions

- The workflow represents all interesting features of a project
- Past projects resemble future projects

Approach

- Automated steps (weekly updates)
  - Obtain workflow from project repositories (VCS/CMS)
    - Typically only the latest state is recorded, the state changes have to be parsed from history log files
    - Obtain job/organization descriptions for individuals
- Interactive steps (web access)
  - Choose suitable past projects and desired transformation
  - Optimize transformation parameters
  - Present predictions based on currently available data

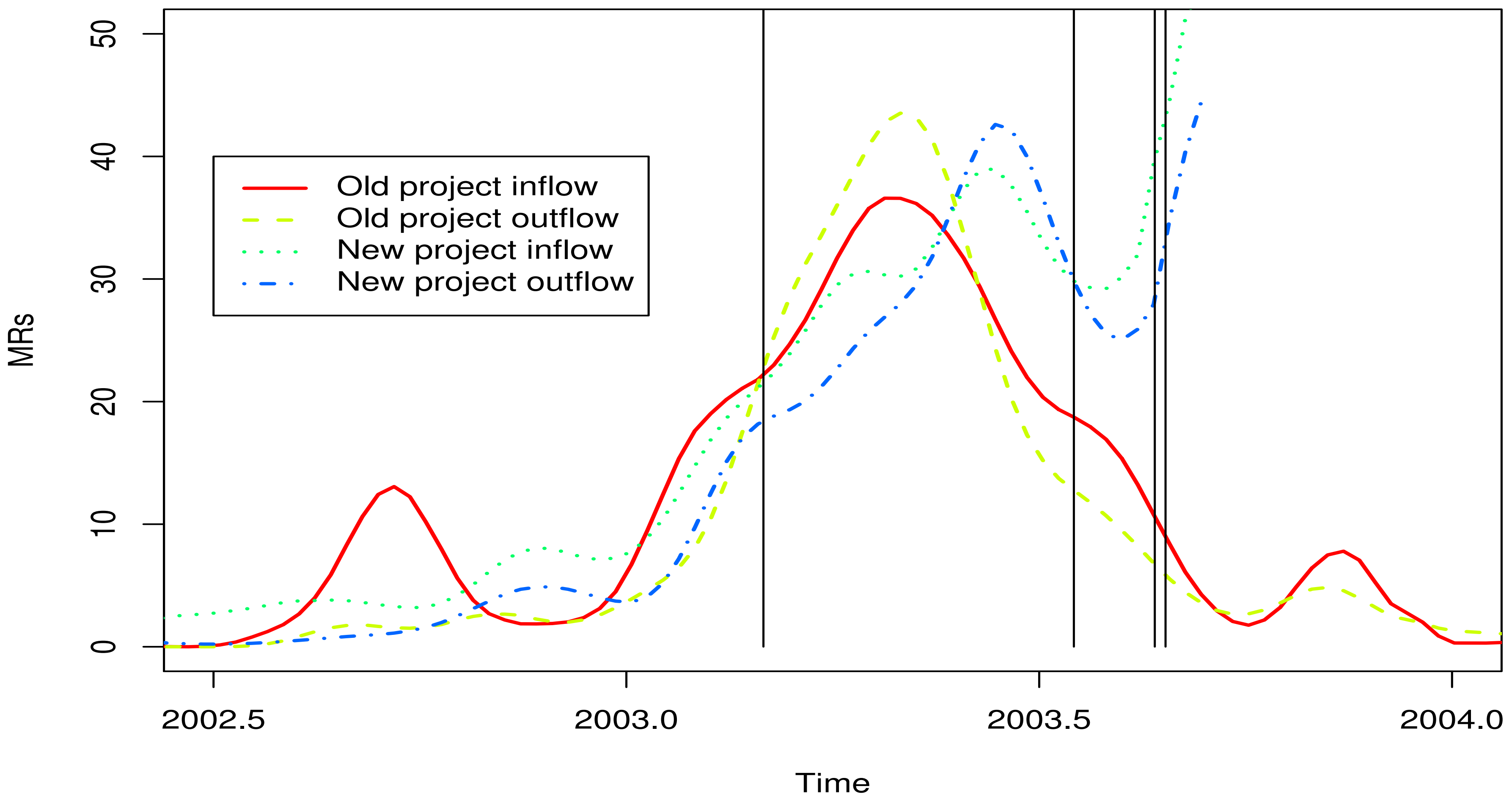
Transformation

- Elicited from project manager
  - Center (start of system test)
  - Staffing (number of people or weighted number)
- Estimated from existing data
  - Scale (small/short incremental or major)
  - Lag (time difference between centers)

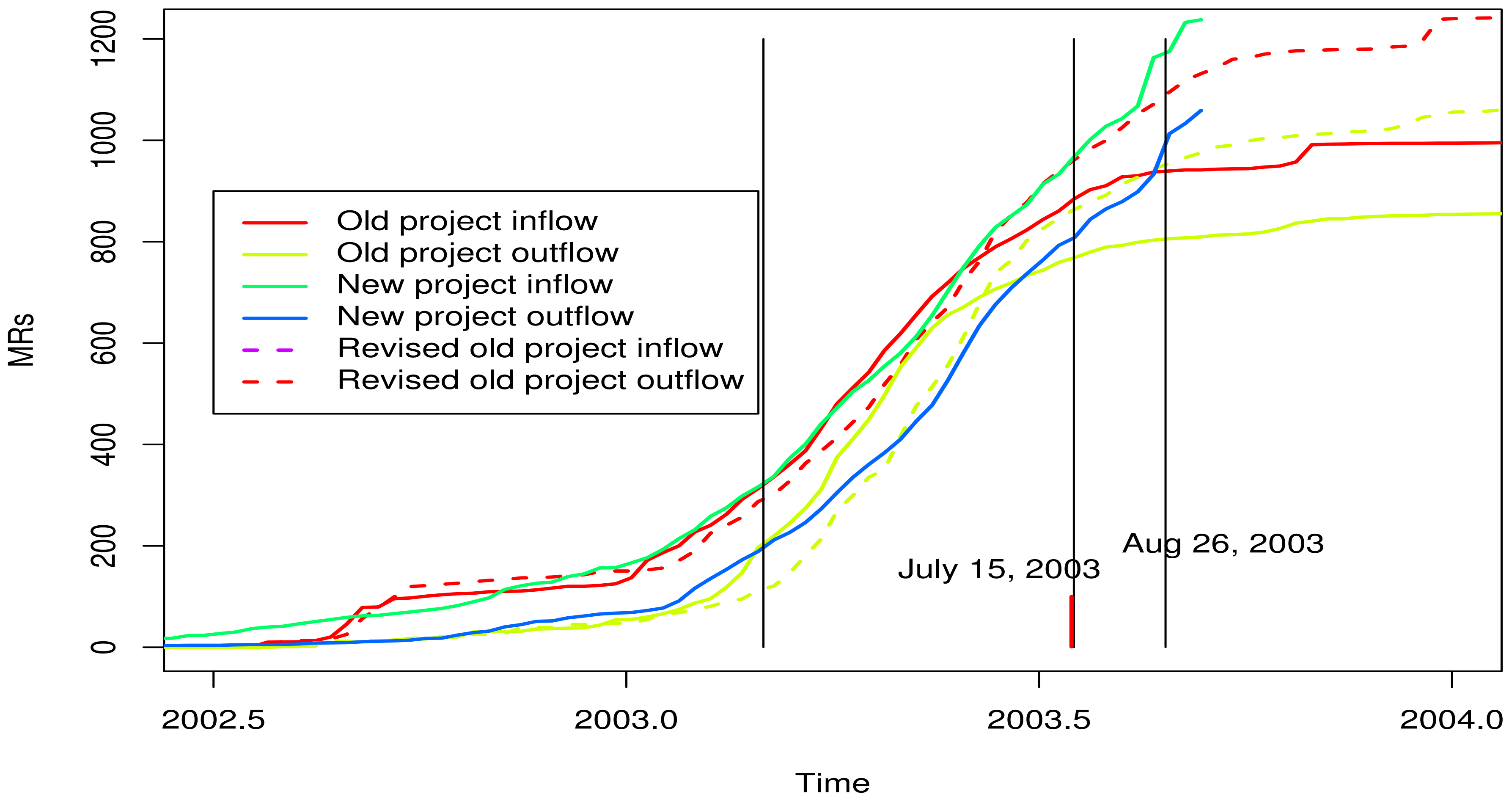
Validation

- Are so few parameters sufficient to capture variations among projects?
- Choosing suitable past release
- Verifying lack of process changes, unanticipated events
- Reduce apparent precision
  - Smooth MR arrival
  - Predict wide ranges
- Evaluating the accuracy of the prediction that was done in March, 2003

Ultimate validation



Ultimate validation



Discussion

- Nonparametric project model
  - Input: selection of past projects and transformations
  - Output: prediction of complete change history for the ongoing project, so any any summary of interest may be predicted
- The accuracy while automatically estimating scale and lag in this case lead to
  - within 3% accurate prediction of defects remaining
  - 5 week or 10% schedule under-estimate
  - 22% underestimate in total incoming defects
  - similar underestimate in total outgoing defects

Discussion

- PT/CM data does represent software project
- Challenges
  - Create suitable models to address problems of practical/theoretical significance
  - Can PT/CM data improved by finding out what information developers would easily and accurately enter in a PT/CM systems?
  - What is the “sufficient statistic” for a software change/project?

References

[1] D. Atkins, T. Ball, T. Graves, and A. Mockus. Using version control data to evaluate the impact of software tools: A case study of the version editor. *IEEE Transactions on Software Engineering*, 28(7):625–637, July 2002.

[2] D. Atkins, A. Mockus, and H. Siy. Measuring technology effects on software change cost. *Bell Labs Technical Journal*, 5(2):7–18, April–June 2000.

[3] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development: Distance and speed. In *23nd International Conference on Software Engineering*, pages 81–90, Toronto, Canada, May 12-19 2001.

[4] Audris Mockus, Roy T. Fielding, and James Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):1–38, July 2002.

[5] Audris Mockus and James Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In *2002 International Conference on Software Engineering*, pages 503–512, Orlando, Florida, May 19-25 2002. ACM Press.

[6] Audris Mockus and David M. Weiss. Predicting risk of software changes. *Bell Labs Technical Journal*, 5(2):169–180, April–June 2000.

[7] Audris Mockus and David M. Weiss. Globalization by chunking: a quantitative approach. *IEEE Software*, 18(2):30–37, March 2001.

[8] Audris Mockus, David M. Weiss, and Ping Zhang. Understanding and predicting effort in software projects. In *2003 International Conference on Software Engineering*, pages 274–284, Portland, Oregon, May 3-10 2003. ACM Press.

Presenter's Bio

Audris Mockus
Avaya Labs Research
233 Mt. Airy Road
Basking Ridge, NJ 07920
ph: +1 908 696 5608, fax:+1 908 696 5402
http://mockus.us, mailto:audris@mockus.org,

Audris Mockus conducts research of complex dynamic systems. He designs data mining methods to summarize and augment the system evolution data, interactive visualization techniques to inspect, present, and control the systems, and statistical models and optimization techniques to understand the systems. Audris Mockus received B.S. and M.S. in Applied Mathematics from Moscow Institute of Physics and Technology in 1988. In 1991 he received M.S. and in 1994 he received Ph.D. in Statistics from Carnegie Mellon University. He works at Software Technology Research Department of Avaya Labs. Previously he worked at Software Production Research Department of Bell Labs.