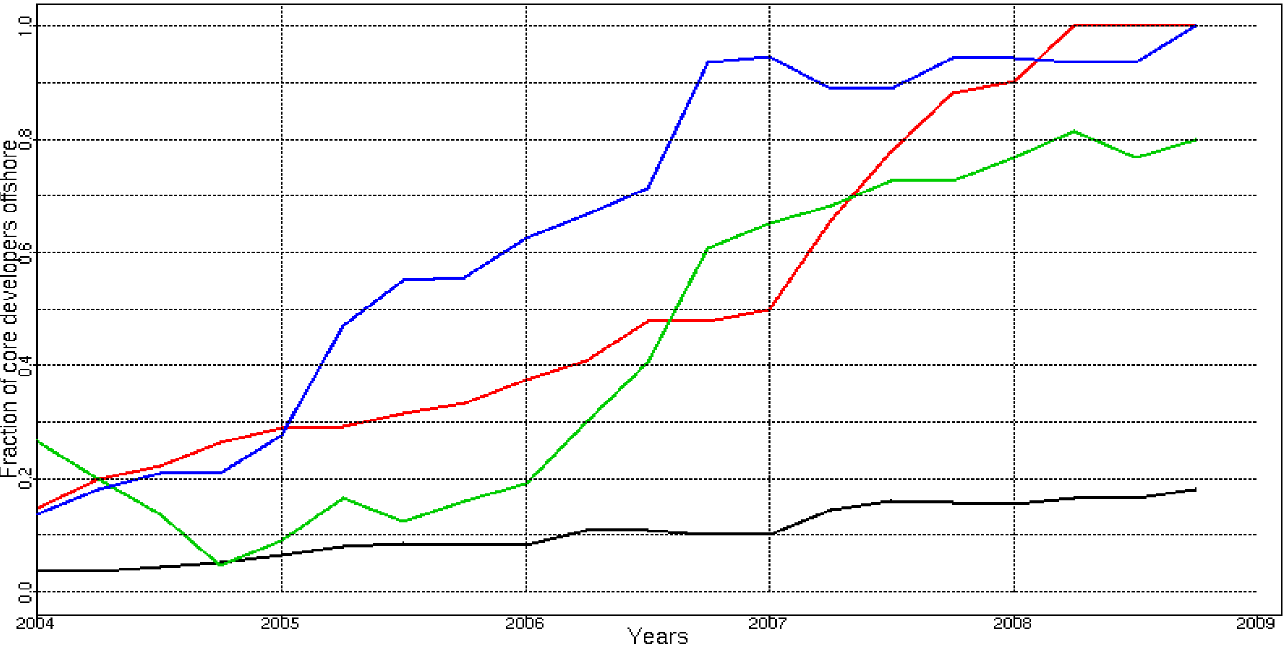
Inverse Conway's law: How product structure shapes organization structure

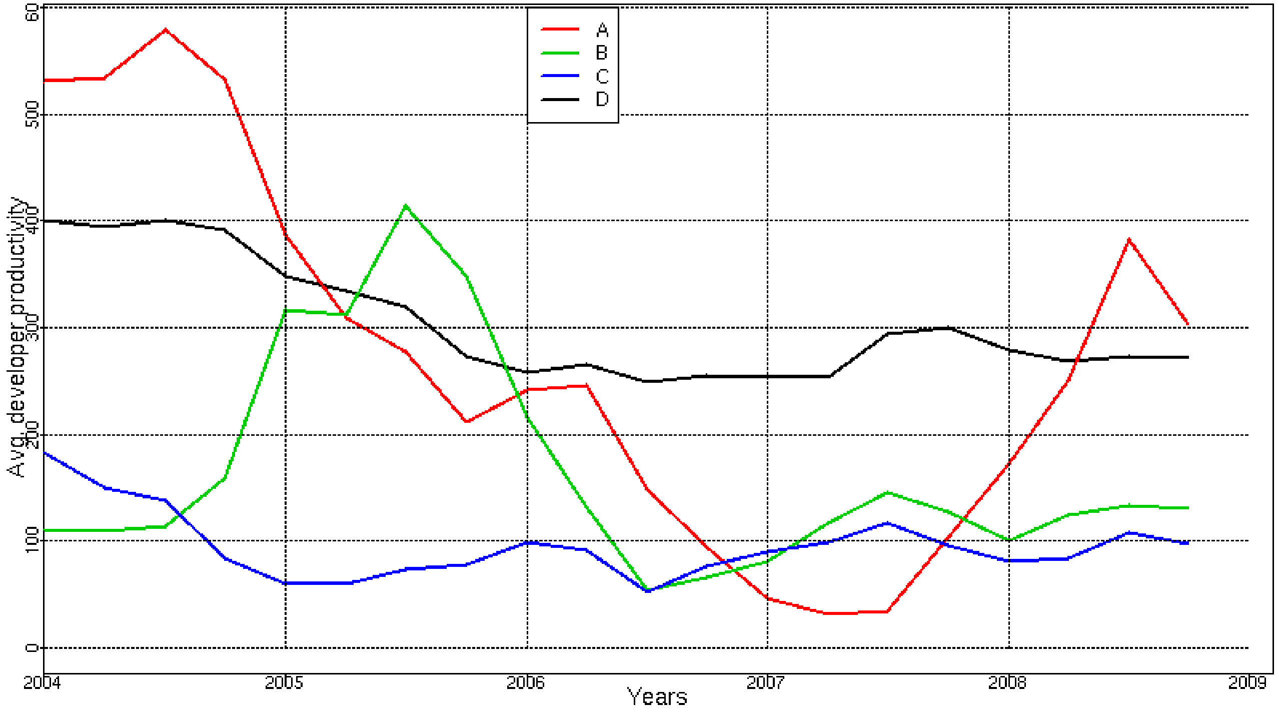
ZHOU, Minghui, zhmh@sei.pku.edu.cn
Peking University
MOCKUS, Audris, audris@avaya.com
WEISS, David, weiss@avaya.com
Avaya Research Labs

2009.5.19

Offshoring story

- Background
- Transfer
- Productivity



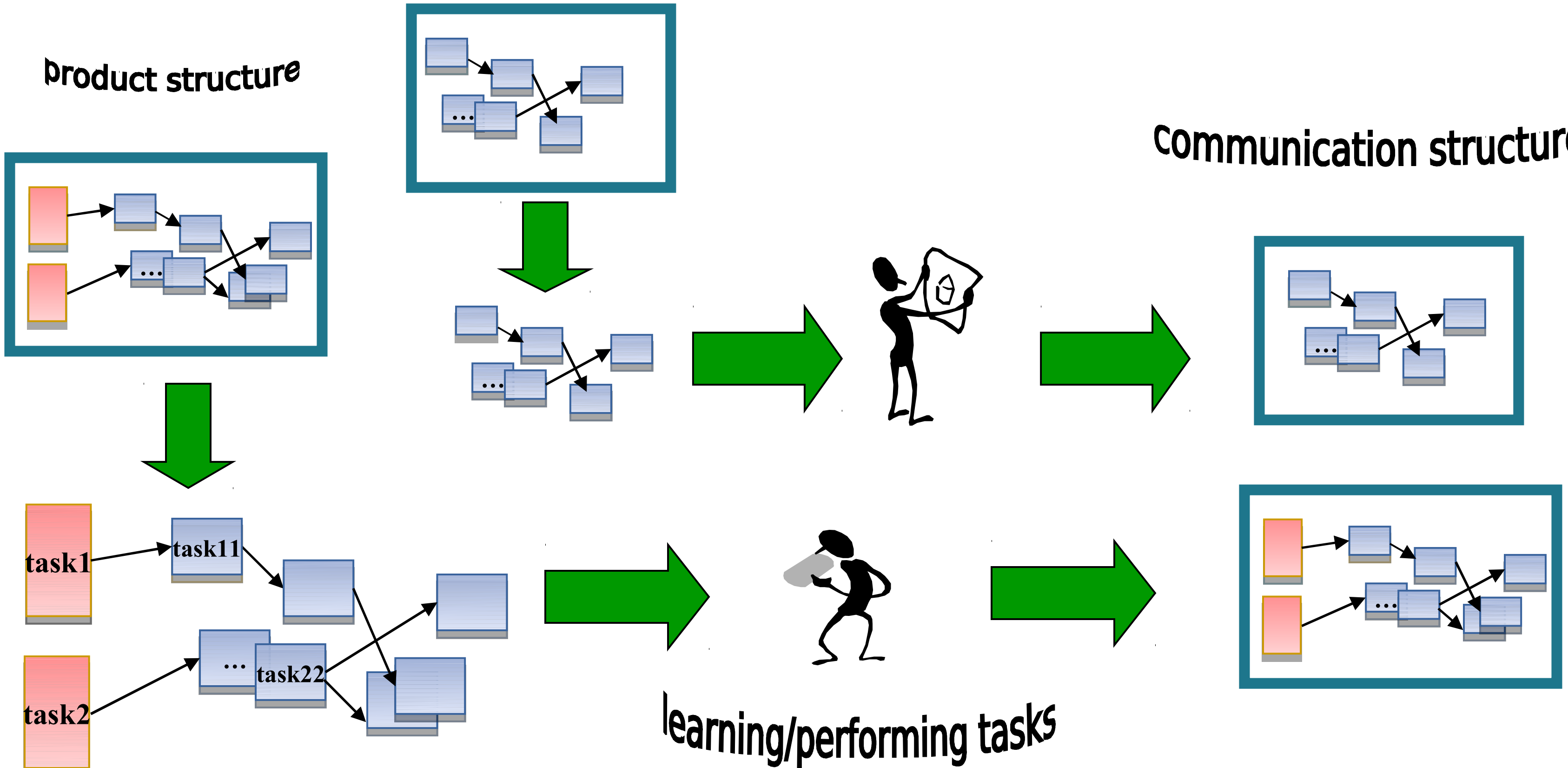


Offshoring indications

- **Developers learn over time**
  - **It took time to master the legacy system**
- **The offshoring happened with little help from original teams**
  - **The main resource for learning is the product itself**
- **Different projects showed different practices**
  - **Products come from different acquired companies, and have different culture embedded in them**
  - **Different learning practices show different culture influences**

Inverse Conway's law

- **Product structure shapes communication structure through learning**
  - ***An existing system shapes the communications of people who maintain or enhance it***
- **Learning is to perform regular project tasks (practice)**
- **Tasks are defined by product structure**
- **Centrality of developers are defined by tasks**
- **Communications are defined by centrality of developers**



Methodology -Qualitative and Quantitative study

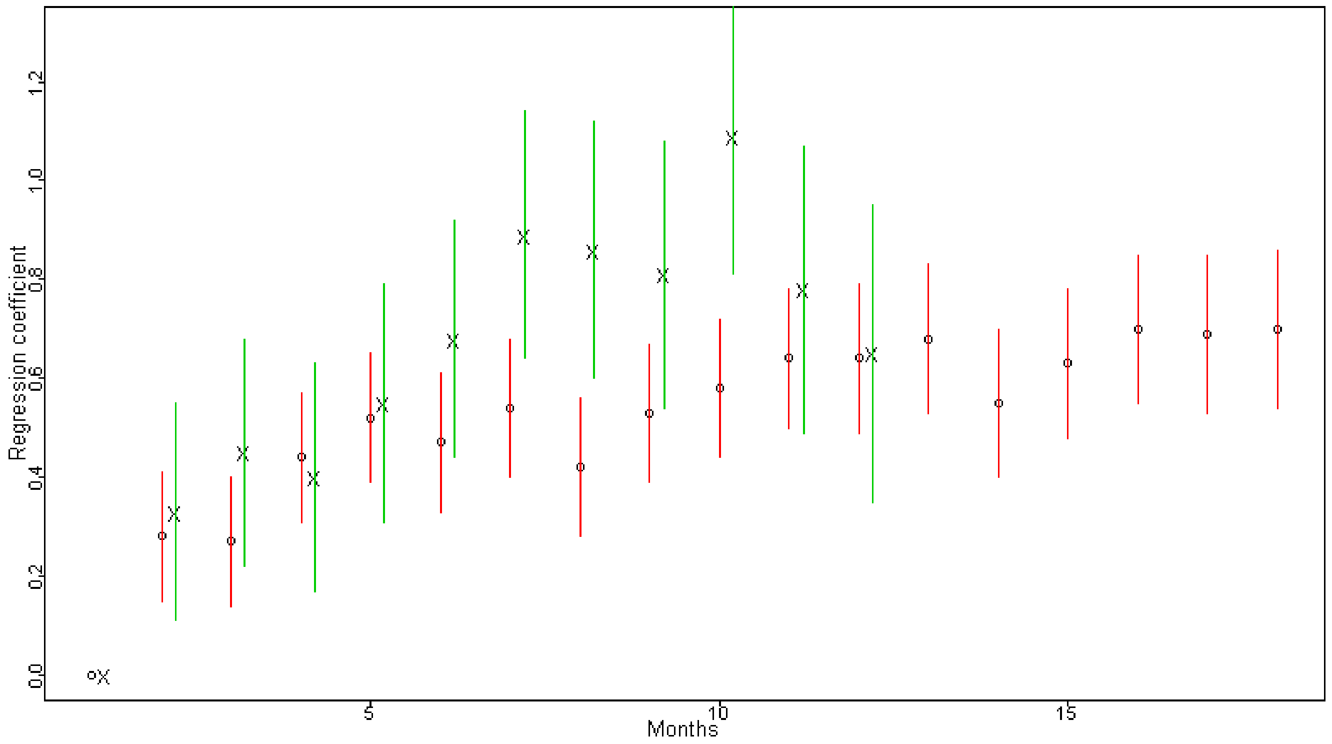
- **Qualitative study**
  - **Designed interview questions, focusing on developers' involvement:**
    - **what to learn when joining, what help was available, what resources they could refer to, how they resolve problems, how they get assignments, who communicated most often, …**
  - **Sampled people for interview: 3 developers per project**
  - **Explained our purpose before interview**
  - **Conducted the interview through telephone conference**
- **Quantitative study**
  - **Access ClearCase and SCCS, Avaya post…**
  - **Filter data**
  - **Every observation is a task-related change, and every change affects a module and is related to a Modification Request (MR)**

tasks

- **Learning *is not to achieve knowledge structure, but the* participation of practice *in the community (LPP approach [3])***
  - ***Developers learn through performing regular project tasks***
- **Regression model: log(productivity+1) ~ ID + tenure + log(practice+1)**
  - **Response: productivity (number of changes per staff-month)**
  - **Predictors: Learning experience**
    - **tenure, i.e., the months from hiring day**
    - **practice, i.e., the changes the developer has made till that month**
  - **All coefficients were significant with R^2 around 0.32**

A-C: green

D: red



Tasks

- Product structure
  - Modules
    - Product package/subsystem
    - Functionality
  - Activities: types of development activity
    - E.g, bug fixing, new feature development. ..
- *Tasks are assigned based on product structure*
  - Work on what module
  - Work through what activity
- Centrality of tasks determines centrality of developers
  - Centrality of a task
    - Customer dimension
    - Long-term impact dimension
    - System-wide impact dimension

different tasks

- **Different roles do different tasks, and have different communications**
  - **Manager, Module owner (close to internals of the product), Tier 4 (close to customer) do different tasks**
- **Seniors and juniors do different tasks, and have different communications**

Centrality/ Product structure	Customer dimension	Long-term impact dimension	System-wide impact dimension
Module structure	"I have worked in almost all areas of C, and am now a technical leader, and responsible for telephony modules" (senior)	"The module changes are reviewed by the experts (seniors) in case they affect the design"	"When I joined I had web client"; "Integration test is given (to novices)"; "Adding printouts to logs" (novice) "I work on voice/XML (browser). I have worked on many modules, because the browser interacts with many modules" (senior)
Activity structure	"I am the contact person for sales demo and data base administration tasks" (senior)	"We would be happy if we get new, interesting features to develop" (offshore)	"Some simple MRs are given" (to novices)

Developers become more central through practice

- **Regression model : log(# of logins+1) ~ ID + log(*practice*+1)**
  - Response: task centrality (how many people have made changes for that module)
  - Predictors: learning experience
    - tenure, i.e., the days from hiring day to the day change was made
    - practice, i.e., the changes the developer has made till that change
  - All coefficients were significant with R^2 around 0.59

- **Other metrics of task centrality**
  - How many changes have been made for that module
  - How many releases are related to that MR
  - If the MR was reported by a customer
  - A non-customer bug fix or new development

When developers practice/learn more, they appear to do more central tasks, in turn, they play more important roles, and have more central communications

	Estimate	Std. Error	t value	Pr(> |t|)
(Intercept)	5.63	0.23	24.14	0.00
log(*practice*+1)	0.05	0.01	9.13	0.00
Developers = 136, Observations=18192 R^2= 0.59				

Best practice?

- **Can there be best software practices?**
  - **The projects differed in ways they provided resources, and ways that developers achieved their skills and implemented common software tasks**
    - **A: *"If we are stuck on a problem, we check out the code to see who changed the code along with the descriptions." "We look through Compas for design documents to understand the component architecture." "If the person is still in the company we ask if they can provide any insights. If not we look at every relevant document in Compass." "If we see more issues we go through QQ to look for similar issues." "We make guesses on keywords to search."***
    - **B: *"In order to locate the bug, we go through all the files; and go through the code to figure out how it works."***
    - **C: *"The first thing is to make a call. I made a call, and dropped it, and looked at traces and logs, to understand what my module did. I gradually added more complexity to scenarios. So I tried to follow code flow."***
  - **Such differences are probably caused by the different origins of each project**
- **It remains to be seen if these practices have been optimized for a particular project or could be improved by borrowing best practices from other projects**

Cultural firewall?

- **Can an organization with a legacy product adapt to changing times?**
  - **The developer practices reproduce in a completely new team through individuals' learning, and individuals learn from legacy artifacts which are imbued with the old culture**
    - *A: "The central repository is on a restricted share point"; "Code is on ClearCase, including the traces showing who changed the code along with the descriptions"; "Compass is convenient to search for design documents to understand the component architecture";"The defect database is used to look for similar issues".*
    - *B: "Documents from US team told how the code and builds were structured, which are the 2rd most important for knowledge transfer"; "Calls and mail support from US team are the 3rd most important for understanding"; "If had questions I first went to the Pune staff (50-70%), and if needed I sent queries to the US team(1%)"*
    - *C: "There is a repository where all (customer) problems are reported; "On ClearCase we check what was changed and who changed it and what files were included in the change"; "Logs tell which problem area to look at. Each log statement has the module name of the originator".*
  - **The software organizations maintaining legacy products are less likely to be able to adjust to changing competitive business environment**
- **Organizations might need to create a cultural firewall between parts of the organization engaged in new and legacy products**

structure?

- **Roles/individuals and communications?**
  - **What are the roles?**
    - **Communication is defined by multiple organization units represented by roles, preferably roles that are far apart**
  - **What are the communications?**
    - **The developers talk synchronously: f2f, phone, im**
    - **Also asynchronously via email (you know who you are talking with)**
    - **Also via artifacts: MRs, documents, code changes (you don’t know who you are talking with)**

- **Is communication structure static or dynamic?**
  - **For an individual, her communications change dynamically when she change her roles, but is it static or dynamic for the whole project CS? Or if defining communication structure from roles, is it static?**

Product as communication

- The entire product and supporting systems act as communications between past and present
  - Opening an MR you don't know who will be assigned to solve it, editing code, you don't know who will be the next to read/change it
  - Problem tracking systems are even more direct representations of past communications (frozen communications)
- Conway's law (and inverse) are just truism
  - Forward: Product structure embeds the communications among people writing/modifying it
  - Inverse: Product affects communications by being the message

References

- [1] Conway, M.E, “How Do Committees Invent?” Datamation, Vol.14, No. 4, Apr. 1968, pp. 28-31.
- [2] J. Van Maanen and E. Schein, “Towards a theory of organizational socialization”, In B. Staw, editor, Research in organizational behavior, volume 1, pp. 209–264. JAI Press, Greenwich, CT, 1979.
- [3] Lave, J., Wenger, E. “Situated Learning. Legitimate Peripheral Participation”, Cambridge University Press. Cambridge. 1991.
- [4] Weinberg, Gerald, “The Psychology of Computer Programming”, Van Nostrand Reinhold Co. New York, NY, USA. 1988.
- [5] Yunwen Ye, Kouichi Kishida, “Toward an understanding of the motivation Open Source Software developers”, Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, May 03-10, 2003, pp .
- [6] Mostrm, J. E., Boustedt, J., Eckerdal, A., McCartney, R., Sanders, K., Thomas, L., and Zander, C. 2008. “Concrete examples of abstraction as manifested in students' transformative experiences”. In Proceeding of the Fourth international Workshop on Computing Education Research (Sydney, Australia, September 06 - 07, 2008). ICER '08. ACM, New York, NY, 125-136.
- [7] A. Mockus and D. Weiss. “Interval quality: Relating customer-perceived quality to process quality”. In 2008 International Conference on Software Engineering, pages 733–740, Leipzig, Germany, May 10–18 2008. ACM Press.
- [8] Audris Mockus. ”Succession: Measuring Transfer of Code and Developer Productivity”. In 2009 International Conference on Software Engineering, to appear.
- [9] A. Mockus. Software support tools and experimental work. In V. Basili and et al, editors, Empirical Software Engineering Issues: Critical Assessments and Future Directions, volume LNCS 4336, pages 91–99. Springer, 2007.
- [10] R. Basili, D. M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, vol. SE-10, no.6, November 1984, pp. 728-738.
- [11] Ikujiro Nonaka, “A dynamic theory of organizational knowledge creation”, Organization Science 5 (1), 1994: 14-37.
- [12] G. von Krogh, S. Spaeth, and K. R. Lakhani, "Community, Joining, and Specialization in Open Source Software Innovation: A Case Study", Research Policy 32(7), July 2003, pp. 1217-1241.
- [13] Ritter, F. E., & Schooler, L. J. “The learning curve”. In International Encyclopedia of the Social and Behavioral Sciences (2002), 8602-8605. Amsterdam: Pergamon
- [14] Andrew Begel and Beth Simon. Novice Software Developers, All Over Again. In the International Computing Education Research Workshop, September 2008. Sydney, Australia.
- [15] A. Mockus and D. M. Weiss. Globalization by chunking: a quantitative approach. IEEE Software, 18(2):30–37, March 2001.
- [16] Cataldo, M., Herbsleb, J.D., Carley, K. “Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity”, 2nd Symposium of Empirical Software Engineering and Measurement, Kaiserslautern, Germany, 2008.
- [17] AJ Ko, R DeLine, G Venolia, "Information Needs in Collocated Software Development Teams", 29th International Conference on Software Engineering, 2007. ICSE 2007, 20-26 May 2007, pp.344-353
- [18] Perry, D. E., N. A. Staudenmayer, and L. G. Votta. People, Organizations, and Process Improvement. IEEE Software. 11, 4, 1994, 36-45.
- [19] Herbsleb, J.D. & Mockus, A. “Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering”, In proceedings, ACM Symposium on the Foundations of Software Engineering (FSE), Helsinki, Finland, 2003, pp. 112-121.