List of Topics for:
Human Aspects of Software Development (HASD)

CMU Course 05-899D

Developed by Thomas LaToza and Brad Myers as a reading list for a course in the Spring of 2011 in the School of Computer Science at Carnegie Mellon University. Additional input from the students in the course, whose names are listed below each topic. (Last updated, April, 2011.). Please let us know if you find this useful, and if there are other articles or topics that should be added.

Table of Contents:

Note: For topics which have good survey articles, the survey paper itself is listed, but not all of the (relevant) papers it reviews. Survey papers are in bold.

1. Introduction to the field

(Edited and Presented by Brad Myers -- see slides of overview presentation)

Brad A. Myers, John F. Pane and Andy Ko, "Natural Programming Languages and Environments". *Communications of the ACM*. (special issue on End-User Development). Sept, 2004, vol. 47, no. 9. pp. 47-52. pdf

Brad A. Myers, David Canfield Smith and Bruce Horn. "Report of the `End-User Programming' Working Group," Languages for Developing User Interfaces, Boston, MA, Jones and Bartlett. 1992. pp. 343-366. -- Gentle-Slope Systems

2. Research Methods

2.1 HCI Research Methods

(Edited and Presented by Thomas LaToza -- see slides of overview presentation)

2.1.1 General

Paul Dourish. (2006). Implications for design. In Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06), Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, 541-550.

Mook, D. (1983). In Defense of External Invalidity. American Psychologist, Vol. 38, No. 4, pp. 379-387.

Suresh K. Bhavnani, Bonnie E. John. Delegation and circumvention: two faces of efficiency. In CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems (1998), pp. 273-280.

Wayne D. Gray and Marilyn C. Salzman. (1998). Damaged merchandise? a review of experiments that compare usability evaluation methods. Hum.-Comput. Interact. 13, 3 (September 1998), 203-261.

Saul Greenberg and Bill Buxton. 2008. Usability evaluation considered harmful (some of the time). In Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI '08). ACM, New York, NY, USA, 111-120.

2.1.2 Conducting HCI studies

Field observations, ethnography, interviews, indirect observations, qualitative methods
Michael Quinn Patton. (2002). Qualitative Research & Evaluation Methods. Sage Publications.
Textbook for HCII research methods class

Natural programming John F. Pane, Chotirat "Ann" Ratanamahatana, and Brad A. Myers, "Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems", *International Journal of Human-Computer Studies (IJHCS)*. Special Issue on Empirical Studies of Programmers, vol. 54, no. 2, February 2001, pp. 237-264.

Contextual inquiry Beyer, H. and Holtzblatt, K. 1997. Contextual Design: Defining Customer-Centered Systems. Morgan Kaufman.
Textbook for HCII HCI methods class

Quantitative data analysis, experiment design, surveys Robert Rosenthal & Ralph Rosnow. (2007). Essentials of Behavioral Research: Methods and Data Analysis. McGraw-Hill.

Surveys
Barbara A. Kitchenham and Shari L. Pfleeger. (2008). Personal Opinion Surveys. In Guide to Advanced Empirical Software Engineering. Springer-Verlag, London.

Don A. Dillman, "Mail and Internet Surveys: The Tailored Design Method", Wiley; 2nd edition (August 18, 2006)

Hartman, J.M., Forsen, J.W., Wallace, M.S., Neely, J.G. (2002). Tutorials in clinical research: Part IV: Recognizing and controlling bias. Laryngoscope, 112, 23-31.

J. Scott Armstrong and Terry S. Overton, "Estimating Non response Bias in Mail Surveys", Journal of Marketing Research, Vol. 14, No. 3, Special Issue: Recent Developments in Survey Research (Aug., 1977), pp. 396-402

Robert A. Ellis, Calvin M. Endo, J. Michael Armer, "The Use of Potential Nonrespondents for Studying Nonresponse Bias", The Pacific Sociological Review, Vol. 13, No. 2 (Spring, 1970), pp. 103-109

Wizard of Oz David Maulsby, Saul Greenberg and Richard Mander. "Prototyping an Intelligent Agent through Wizard of Oz," *Human Factors in Computing Systems,* Proceedings INTERCHI'93. Amsterdam, The Netherlands, Apr, 1993. pp. 277-284.

Sketching and Prototyping Bill Buxton. 2007. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Heuristic evaluation *Nielsen, J., Enhancing the explanatory power of usability heuristics, CHI'94 Conference Proceedings, (1994).*

Cognitive walkthrough C. Wharton et al. "The cognitive walkthrough method: a practitioner's guide" in J. Nielsen & R. Mack "Usability Inspection Methods" pp. 105-140.

Cognitive dimensions of notations Thomas R. G. Green, Marian Petre. (1996). Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. J. Vis. Lang. Comput. 7(2): 131-174.

2.2 Research Methods for Studies of Developers

(Edited and Presented by Thomas LaToza -- see slides of overview presentation)

Forrest Shull, Janice Singer, Dag I.K. Sjoberg (eds). (2008). Guide to Advanced Empirical Software Engineering. Springer-Verlag, London.

Tim Menzies and Forrest Shull. (2011). The quest for convincing evidence. In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 3-16.

Lutz Prechelt & Marian Petre. (2011). Credibility, or why should I insist on being convinced? In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 17-34.

Andrew Ko. (2011). Understanding software engineering through qualitative methods. In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 55-63.

Carolyn B. Seaman. (1999). Qualitative Methods in Empirical Studies of Software Engineering. In Transactions on Software Engineering, 25(4).

D. I. K. Sjoeberg, J. E. Hannay, O. Hansen, et al. (03 September 2005). A survey of controlled experiments in software engineering. IEEE Transactions on Software Engineering, Vol. 31, No. 9. pp. 733-753.

B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, et al.. Preliminary guidelines for empirical research in software engineering. Software Engineering, IEEE Transactions on, Vol. 28, No. 8. (2002), pp. 721-734.

Jorge Aranda, Gina Venolia: The secret life of bugs: Going past the errors and omissions in

software repositories. ICSE 2009: 298-308.

Jonathan Lung, Jorge Aranda, Steve M. Easterbrook, and Gregory V. Wilson. 2008. On the difficulty of replicating human subjects studies in software engineering. In Proceedings of the 30th international conference on Software engineering (ICSE '08). ACM, New York, NY, USA, 191-200.

Jo E. Hannay, Dag I. K. Sjoberg, and Tore Dyba. 2007. A Systematic Review of Theory Use in Software Engineering Experiments. IEEE Trans. Softw. Eng. 33, 2 (February 2007), 87-107.

3. Learning to program

3.1 Learning to program, how people naturally think about computation

(To Be Edited and Presented by Cyrus Omar -- see slides of overview presentation)

Bonar, J., and Soloway, E.. 1983. Uncovering principles of novice programming. In Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (POPL '83). ACM, New York, NY, USA, 10-13.

Miller, L. A. 1981. "Natural language programming: Styles, strategies, and contrasts." IBM Systems Journal 29(2): 184–215.

du Boulay, B. 1989. Some difficulties of learning to program. In E. Soloway & J. C. Spohrer, Eds., Studying the Novice Programmer, pp. 283-299. Hillsdale, NJ: Lawrence Erlbaum Associates.

Galotti, K.M. and W.F. Ganong, III. 1985. What Non-Programmers Know About Programming: Natural Language Procedure Specification. International Journal of Man-Machine Studies 22: 1-10.

Bruckman, A. and Edwards, E. 1999. Should We Leverage Natural-Language Knowledge? Proceedings of CHI 99. New York: ACM Press, pp. 207-214.

Mark Guzdial. (2011). Why is it so hard to learn to program? In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 111-121.

John Pane and Brad Myers. *Usability Issues in the Design of Novice Programming Systems*, Carnegie Mellon University School of Computer Science Technical Report CMU-CS-96-132. and Human Computer Interaction Institute Technical Report

CMU-HCII-96-101, August, 1996.

John F. Pane, Chotirat "Ann" Ratanamahatana, and Brad A. Myers, "Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems", *International Journal of Human-Computer Studies (IJHCS)*. Special Issue on Empirical Studies of Programmers, vol. 54, no. 2, February 2001, pp. 237-264.

John Pane and Brad Myers, "Tabular and Textual Methods for Selecting Objects from a Group," *IEEE Symposium on Visual Languages*, VL'2000, Seattle, Washington, September 10-14, 2000. pp. 157-164.

Judith Good, Katy Howland, Keiron Nicholson, "Young People's Descriptions of Computational Rules in Role-Playing Games: An Empirical Study," Visual Languages and Human-Centric Computing, IEEE Symposium on, pp. 67-74, 2010 IEEE Symposium on Visual Languages and Human-Centric Computing, 2010.

Gary Lewandowski, Dennis J. Bouvier, Robert McCartney, Kate Sanders, and Beth Simon. “Commonsense computing (episode 3): concurrency and concert tickets”. In Proceedings *of the third international workshop on Computing education research (ICER '07)*. ACM, New York, NY, USA, 133-144. ACM DL, pdf

Mayer, Richard E., The Psychology of How Novices Learn Computer Programming. ACM Computing Surveys, 1981. 13(1): p. 121-141.

4. Design decisions

4.1 Design, designing with diagrams

(To Be Edited and Presented by Chris Martens -- see slides of overview presentation)

Tom Moran & John Carroll (Eds). *Design Rationale: Concepts, Techniques, and Use*. 1996. *Thomas has this book.*

Herbsleb, J. D., Klein, H., Olson, G. M., Brunner, H., Olson, J. S., & Harding, J. (1995). Object-oriented analysis and design in software project teams. Human Computer Interaction, 10, 249-292.

Herbsleb, J. D. (1999). Metaphorical representation in collaborative software engineering. In Proceedings, International Joint Conference on Work Activities, Coordination, and Collaboration, San Francisco, CA, February 22-25, pp. 117-125. (pdf)

Marian Petre and Alan F. Blackwell. 1999. Mental imagery in program design and visual

programming. Int. J. Hum.-Comput. Stud. 51, 1 (July 1999), 7-30.

Cherubini, M., Venolia, G. DeLine, R. and Ko, A. J. (2007). Let's Go to the Whiteboard: How and Why Software Developers Draw Code. ACM Conference on Human Factors in Computing Systems (CHI).

Mauro Cherubini, Gina Venolia, and Rob DeLine, Building an Ecologically-valid, Large-scale Diagram to Help Developers Stay Oriented in Their Code, in VLHCC '07: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing, IEEE Computer Society, Washington, DC, USA, September 2007.

Koji Yatani, Eunyoung Chung, Carlos Jensen, and Khai N. Truong. 2009. Understanding how and why open source contributors use diagrams in the development of Ubuntu. In Proceedings of the 27th international conference on Human factors in computing systems (CHI '09). ACM, New York, NY, USA, 995-1004.

Uri Dekel and James D. Herbsleb, "Notation and Representation in Collaborative Object-Oriented Design", ACM International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'07), October 2007.

Moody, D. (2009). The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. In Transactions on Software Engineering, 35 (6), 756-779.

Helen C. Purchase, Ray Welland, Matthew McGill, Linda Colpoys. (2004). Comprehension of diagram syntax: an empirical study of entity relationship notations. *Int. J. Hum.-Comput. Stud*. 61(2): 187-203. pdf

Eunyoung Chung, Carlos Jensen, Koji Yatani, Victor Kuechler, Khai N. Truong. (2010). "Sketching and Drawing in the Design of Open Source Software," Visual Languages and Human-Centric Computing, IEEE Symposium on, pp. 195-202, 2010 IEEE Symposium on Visual Languages and Human-Centric Computing.

4.2 Coordinating developers

(Edited and Presented by Jim Herbsleb -- see slides of overview presentation)

4.2.1 Collocated teams

Bill Curtis, Herb Krasner, and Neil Iscoe. 1988. A field study of the software design process for large systems. Commun. ACM 31, 11 (November 1988), 1268-1287.

Robert E. Kraut and Lynn A. Streeter. 1995. Coordination in software development. Commun. ACM 38, 3 (March 1995), 69-81.

Teasley, S. D., Covi, L. A., Krishnan, M. S. and Olson, J. S. Rapid Software Development through Team Collocation. *IEEE Transactions on Software Engineering*, 28, 7 2002), 671-683. IEEE DL or ACM DL or pdf

Cleidson R. Souza and David F. Redmiles. 2009. On The Roles of APIs in the Coordination of Collaborative Software Development. Comput. Supported Coop. Work 18, 5-6 (December 2009), 445-475.

4.2.2 Global software development

James D. Herbsleb. 2007. Global Software Engineering: The Future of Socio-technical Coordination. In 2007 Future of Software Engineering (FOSE '07). IEEE Computer Society, Washington, DC, USA, 188-198.

Herbsleb, J. D., & Grinter, R. E. (1999). Architectures, coordination, and distance: Conway's Law and beyond. IEEE Software, Sept/Oct 1999, 63-70. (pdf)

Herbsleb, J.D., Mockus, A., Finholt, T.A., & Grinter, R.E. (2000). Distance, dependencies, and delay in a global collaboration. InProceedings, ACM Conference on Computer-Supported Cooperative Work, Philadelphia, PA, Dec. 2-7, pp. 319-328. (pdf)

Mockus, A., Fielding, R., & Herbsleb, J.D. (2002). Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11, 3, pp. 309-346. (pdf)

Herbsleb, J.D. & Mockus, A. (2003). An empirical study of speed and communication in globally-distributed software development. IEEE Transactions on Software Engineering, 29, 3, 1-14. (pdf)

J. D. Herbsleb, D. J. Paulish, and M. Bass. Global software development at siemens: experience from nine projects. In Proceedings of the 27th International Conference on Software Engineering, pages 524–533. ACM, 2005.

Gurbani, V.K., Garvert, A., & Herbsleb, J.D. (2006). A Case study of a corporate open source development model. In Proceedings of the International Conference on Software Engineering, Shanghai, China, May 20-25, pp. 472-481. (pdf)

Espinosa, J. A., Slaughter, S. A., Kraut, R. E., & Herbsleb, J. D. (2007). Team Knowledge and Coordination in Geographically Distributed Software Development. Journal of Management Information Systems, 24, 1, pp. 5 – 12. (pdf)

Espinosa, A., Slaughter, S., Kraut, R., & Herbsleb, J. (2007). Familiarity, Complexity and Team Performance in Geographically Distributed Software Development. Organization Science,

July-August,18, pp. 613 – 630. (pdf)

W. F. Boh, S. A. Slaughter, and J. A. Espinosa, "Learning from Experience in Software Development: A Multilevel Analysis," Management Science, vol. 53, pp. 1315-1331, 2007.

Cleidson R. B. de Souza, David F. Redmiles: An empirical study of software developers' management of dependencies and changes. ICSE 2008: 241-250

Bird, C., Nagappan, N., Devanbu, P.T., Gall, H., and Murphy, B. Does distributed development affect software quality? An empirical case study of Windows Vista. In Proceedings of the 31st International Conference on Software Engineering (2009), 518-528.

Cataldo, M., Mockus, A., Roberts, J.A., & Herbsleb, J.D. (2009). Software Dependencies, Work Dependencies, and Their Impact on Failures. IEEE Transactions on Software Engineering, 99, 864-878. (pdf)

M. Cataldo and S. Nambiar, "On the Relationship between Process Maturity and Geographic Distribution: An Empirical Analysis of Their Impact on Software Quality," FSE 2009.

N. Ramasubbu, M. Cataldo, R.K Balan, & J.D. Herbsleb. "Configuring Global Software Teams: A Multi-Company Analysis of Project Productivity, Quality, and Profits." To appear, ICSE 2011.

4.3 Onboarding, Finding experts

(To Be Edited and Presented by Kun Niu -- see slides of overview presentation)

Thomas Fritz, Jingwen Ou, Gail C. Murphy, and Emerson Murphy-Hill. 2010. A degree-of-knowledge model to capture source code familiarity. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10), Vol. 1. ACM, New York, NY, USA, 385-394.

Mockus, A. and Herbsleb, J.D. Expertise browser: a quantitative approach to identifying expertise. In Proceedings of the 22rd International Conference on Software Engineering (2002), 503-512.

David Ma, David Schuler, Thomas Zimmermann, Jonathan Sillito. Expert Recommendation with Usage Expertise. In Proceedings of ICSM (Short Paper) 2009.

Barthélémy Dagenais, Harold Ossher, Rachel K.E. Bellamy, Martin P. Robillard, and Jacqueline P. de Vries. Moving into a New Software Project Landscape. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, pages 275-284, May 2010.

Andrew Begel and Beth Simon. 2008. Struggles of new college graduates in their first software

development job. In Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08). ACM, New York, NY, USA, 226-230.

Thomas Fritz and Gail C. Murphy. 2010. Using information fragments to answer the questions developers ask. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10), Vol. 1. ACM, New York, NY, USA, 175-184.

Suzanne P. Mikawa, Sharon K. Cunnington, Scott A. Gaskins, Removing barriers to trust in distributed teams: understanding cultural differences and strengthening social ties IWIC '09 Proceeding of the 2009 international workshop on Intercultural collaboration

4.4 Maintaining awareness

Olga Kulyk, Gerrit van der Veer, Betsy van Dijk Situational awareness support to enhance teamwork in collaborative environments ECCE '08: Proceedings of the 15th European conference on Cognitive ergonomics: the ergonomics of cool interaction, January 2008

Reid Holmes and Andrew Begel. Deep Intellisense: A Tool for Rehydrating Evaporated Information. In Proceedings of the Working Conference on Mining Software Repositories, 2008. 23-26.

Andrew Begel, Yit Phang Khoo, Thomas Zimmermann: Codebook: discovering and exploiting relationships in software repositories. ICSE2010: 125-134.

Emerson Murphy-Hill and Gail Murphy. Peer Interaction Effectively, yet Infrequently, Enables Programmers to Discover New Tools. Computer Supported Cooperative Work, 2011.

Biehl, J. T., Czerwinski, M., Smith, G., and Robertson, G. G. 2007. Fastdash: a visual dashboard for fostering awareness in software teams. In Proceedings of the SIGCHI conference on Human factors in computing systems. CHI '07. ACM, New York, NY, USA, 1313{1322.

A. Sarma, D. Redmiles, and A. van der Hoek, Empirical Evidence of the Benefits of Workspace Awareness in Software Configuration Management, ACM SIGSOFT International Symposium on the Foundations of Software Engineering, (FSE 16), Atlanta, Georgia, November 2008, pages 113-123.

de Souza, C.R.B., Redmiles, D.F. The Awareness Network: Should I display my actions to whom? And, whose actions should I monitor?, The 10th European Conference on Computer Supported Co-operative Work (ECSCW 2007, Limerick, Ireland), September 2007, pp. 99-117.

Sarma, A., Redmiles, D., van der Hoek, A. A Comprehensive Evaluation of Workspace

Awareness in Software Configuration Management Systems, IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007, Coeur d'Alne, Idaho), September 2007, pp. 23-26.

A.Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, Tesseract: Interactive Visual Exploration of Socio-Technical Relationships in Software Development,Proceedings of the Thirty-first International Conference on Software Engineering, Vancouver, Canada, May 2009,pages:23-33.

Halverson, C. Ellis, J., Danis, C. and Kellogg, W. Designing task visualizations to support the coordination of work in software development. Proc. ACM CSCW (2006), 39-48.

Gutwin, C., Penner, R., and Schneider, K. 2004. Group awareness in distributed software development. In CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work. ACM, New York, NY, USA, 72-81.

Christoph Treude and Margaret-Anne Storey. 2010. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10), Vol. 1. ACM, New York, NY, USA, 365-374.

Inah Omoronyia, John Ferguson, Marc Roper and Murray Wood. Using Developer Activity Data to Enhance Awareness during Collaborative Software Development. Computer Supported Cooperative Work (2009) 18:509–55.

Sarma, A., Noroozi, Z., and van der Hoek, A. 2003. Palantr: raising awareness among conguration management workspaces. In ICSE '03: Proceedings of the 25th International Conference
on Software Engineering. IEEE Computer Society, Washington, DC, USA, 444{454.

M. Cameron Jones and Elizabeth F. Churchill. 2009. Conversations in developer communities: a preliminary analysis of the yahoo! pipes community. In Proceedings of the fourth international conference on Communities and technologies (C\&\#38;T '09). ACM, New York, NY, USA, 195-204.

Cubranic, D.; Murphy, G.C.; Singer, J.; Booth, K.S.; , "Hipikat: a project memory for software development," *Software Engineering, IEEE Transactions on* , vol.31, no.6, pp. 446- 465, June 2005.

5. Reuse

5.1 Finding code to reuse

(Edited and Presented by Kerry Chang -- see slides of overview presentation)

Yunwen Ye and Gerhard Fischer. 2002. Supporting reuse by delivering task-relevant and personalized information. In Proceedings of the 24th International Conference on Software Engineering (ICSE '02). ACM, New York, NY, USA, 513-523.

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, Scott R. Klemmer. Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code. CHI: ACM Conference on Human Factors in Computing Systems, Boston, MA, 2009.

Mary Beth Rosson and John M. Carroll. 1996. The reuse of uses in Smalltalk programming. ACM Trans. Comput.-Hum. Interact. 3, 3 (September 1996), 219-253.

Joel Brandt, Mira Dontcheva, Marcos Weskamp, Scott R. Klemmer. Example-Centric Programming: Integrating Web Search into the Development Environment. CHI: ACM Conference on Human Factors in Computing Systems, Atlanta, GA, 2010. (try it out) (video).

George Fairbanks, William Scherlis and David Garlan. Design Fragments Make Using Frameworks Easier. In Proceedings of *ACM SIGPLAN Conference on Object Oriented Programs, Systems, Languages, and Applications (OOPSLA),* October 2006. local pdf

A. Sen, "The role of opportunism in the software design reuse process," IEEE Trans. Softw. Eng., vol. 23, no. 7, pp. 418–436, 1997.

D. Kirk, M. Roper, and M. Wood. Identifying and Addressing Problems in Object-Oriented Framework Reuse. Empirical Software Engineering, 12(3):243–274, 2006.

Mathew Mooty, Andrew Faulring, Jeffrey Stylos, Brad A. Myers. Calcite: Completing Code Completion for Constructors Using Crowds. VL/HCC 2010: 15-22.

Reid Holmes and Robert J. Walker. Supporting the investigation and planning of pragmatic reuse tasks. In Proceedings of the International Conference on Software Engineering (Minneapolis, MN, USA. May 25, 2007). ICSE '07.

Steven P. Reiss: Semantics-based code search. ICSE 2009: 243-253.

David Mandelin, Lin Xu, Rastislav Bodik, and Doug Kimelman. 2005. Jungloid mining: helping to navigate the API jungle. In Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation (PLDI '05). ACM, New York, NY, USA, 48-61.

Sahavechaphan, N. and Claypool, K. 2006, “XSnippet: mining For sample code”, Proc. of the 2006 OOPSLA Conference, ACM, 2006, pp. 413-430.

Suresh Thummalapenta and Tao Xie. 2007. Parseweb: a programmer assistant for reusing open source code on the web. In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE '07). ACM, New York, NY, USA, 204-213.

Raphael Hoffmann, James Fogarty, and Daniel S. Weld. 2007. Assieme: finding and leveraging implicit references in a web search interface for programmers. In Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07). ACM, New York, NY, USA, 13-22.

Hartmann, Björn, Leslie Wu, Kevin Collins and Scott R. Klemmer. Programming by a Sample: Rapidly Creating Web Applications with d.mix. In Proceedings of uist 2007: ACM Symposium on User Interface Software and Technology. Newport, Rhode Island, USA, 2007.

Jeffrey Stylos and Brad A. Myers. "Mica: A Programming Web-Search Aid". *2006 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC'06*. Sept 4-8, 2006, Brighton, UK. pp. 195-202. pdf

5.2 Designing & documenting APIs

(Edited and Presented by Brad Myers -- see slides of overview presentation)

5.2.1 Studies of APIs

See: http://sites.google.com/site/apiusability/

Steven Clarke. (2011). How usable are your APIs? In *Making Software: What really works and why we believe it*, Andy Oram and Greg Wilson (eds), 545 - 565.

Jeffrey Stylos, PhD Dissertation, May, 2009, *Making APIs More Usable with Improved API Designs, Documentation and Tools*, CMU-CS-09-130. 171 pages. pdf

Steven Clarke, “Measuring Api Usability.” *Dr. Dobb's Journal; Special Windows/.NET Supplement*, 2004. pp. S6-S9. link

Jeffrey Stylos, Steven Clarke: Usability Implications of Requiring Parameters in Objects' Constructors.*ICSE* 2007: 529-539. ACM DL

Jeffrey Stylos, Brad A. Myers: The implications of method placement on API learnability. *SIGSOFT FSE,* 2008: 105-112. local pdf or ACM version

Jeffrey Stylos, Benjamin Graf, Daniela K. Busse, Carsten Ziegler, Ralf Ehret, Jan Karstens: A

case study of API redesign for improved usability. *VL/HCC 2008*: 189-192. IEEE DL

Jack Beaton, Sae Young Jeong, Yingyu Xie, Jeffrey Stylos, Brad A. Myers: Usability challenges for enterprise service-oriented architecture APIs. *VL/HCC 2008*: 193-196. pdf

Brian Ellis, Jeffrey Stylos, Brad A. Myers: The Factory Pattern in API Design: A Usability Evaluation. *ICSE 2007*: 302-312. ACM or local pdf

Daniel S. Eisenberg, Jeffrey Stylos, Brad A. Myers: Apatite: a new interface for exploring APIs. *CHI 2010*: 1331-1334. local pdf and local movie

Martin P. Robillard. What Makes APIs Hard to Learn? Answers from Developers. *IEEE Software* - Special Issue on the Collaborative and Human Aspects of Software Engineering, 26(6):27-34, November/December 2009. pdf

Martin P. Robillard and Robert DeLine. A Field Study of API Learning Obstacles. *Empirical Software Engineering*, Springer: 2011. To appear. pdf

Jeffrey Stylos and Brad Myers, "Mapping the Space of API Design Decisions," *2007 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC'07*. Sept 23-27, 2007, Coeur d'Alene, Idaho. pp. 50-57. pdf

5.2.2 Tools for Documenting and Helping with APIs

Sae Young Jeong, Yingyu Xie, Jack Beaton, Brad A. Myers, Jeffrey Stylos, Ralf Ehret, Jan Karstens, Arkin Efeoglu, Daniela K. Busse: Improving Documentation for eSOA APIs through User Studies.*IS-EUD 2009*: 86-105. local pdf

Daniel S. Eisenberg, Jeffrey Stylos, Andrew Faulring, Brad A. Myers: Using Association Metrics to Help Users Navigate API Documentation. VL/HCC 2010: 23-30. IEEE DL or local pdf

Jeffrey Stylos, Andrew Faulring, Zizhuang Yang, Brad A. Myers: Improving API documentation using API usage information. *VL/HCC 2009*: 119-126. IEEE DL pdf or local pdf

Max Goldman and Robert C. Miller. "Codetrail: Connecting Source Code and Web Resources." *VL/HCC 2008*, pp. 65-72. local pdf

Uri Dekel and James D. Herbsleb, Improving API Documentation Usability with Knowledge Pushing, *ACM/IEEE International Conference of Software Engineering (ICSE '09)*, May 2009. local pdf

Mathew Mooty, Andrew Faulring, Jeffrey Stylos and Brad Myers. "Calcite: Completing Code Completion for Constructors using Crowds," *2010 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'10)*, Leganés-Madrid, Spain, 21-25 September 2010. pp.

15-22. IEEE DL or local pdf

6. Finding and Understanding Code

6.1 Exploring code

(Edited and Presented by Thomas LaToza -- see slides of overview presentation)

6.1.1 Questions & strategies

Andrew J. Ko, Robert DeLine, and Gina Venolia, Information Needs in Collocated Software Development Teams, in ICSE '07: Proceedings of the 29th international conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, May 2007.

Jonathan Sillito, Gail C. Murphy and Kris De Volder. Asking and Answering Questions during a Programming Change Task. In IEEE Transactions on Software Engineering. 2008.

Thomas D. LaToza and Brad. A. Myers. (2010). Hard to answer questions about code. In the PLATEAU workshop at SPLASH '10.

Abi-Antoun, M., Ammar, N., LaToza, T. (2010). Questions about object structure during coding activities. In the Workshop on Cooperative and Human Aspects of Software Engineering at ICSE '10.

LaToza, T. D., & Myers, B. A. (2010). On the importance of understanding the strategies that developers use. In the Workshop on Cooperative and Human Aspects of Software Engineering at ICSE '10.

Martin P. Robillard, Wesley Coelho, and Gail C. Murphy. How Effective Developers Investigate Source Code: An Exploratory Study. IEEE Transactions on Software Engineering, 30(12):889-903, December 2004.

Thomas D. LaToza, David Garlan, James D. Herbsleb, and Brad A. Myers. 2007. Program comprehension as fact finding. In Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC-FSE '07). ACM, New York, NY, USA, 361-370.

Jamie Starke, Chris Luce, Jonathan Sillito. Searching and Skimming: An Exploratory Study In Proceedings of ICSM 2009.

6.1.2 Traversing structural relationships in code

Thomas D. LaToza and Brad A. Myers. 2010. Developers ask reachability questions. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering -

Volume 1(ICSE '10), Vol. 1. ACM, New York, NY, USA, 185-194.

de Alwis, Murphy and Robillard. A comparative study of three program exploration tools. I*CPC 2007*.

Margaret-Anne D. Storey, Frank D. Fracchia, and Hausi A. M¨uller. Cognitive Design Elements to Support the Construction of a Mental Model During Software Exploration. J. Systems & Software, 44(3), 1999

B de Alwis, GC Murphy (2008). Answering Conceptual Queries with Ferret. In Proceedings of the International Conference on Software Engineering (ICSE), Leipzig, Germany.

D. Janzen and K. D. Volder. Navigating and querying code without getting lost. In Proc. Int'l Conf. Aspect-Oriented Softw. Development, pages 178–187. ACM, 2003.

Michael Würsch, Giacomo Ghezzi, Gerald Reif, and Harald C. Gall. 2010. Supporting developers with natural language queries. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10), Vol. 1. ACM, New York, NY, USA, 165-174.

Anderson, P.; Reps, T.; Teitelbaum, T.; , "Design and implementation of a fine-grained software inspection tool," *Software Engineering, IEEE Transactions on* , vol.29, no.8, pp. 721- 733, Aug. 2003.

6.1.3 Feature location

Norman Wilde, Michelle Bueckellew, Henry Page, Vaclav Rajlich, LaTreva Pounds. (2003). A comparison of methods for locating features in legacy software. Journal of Systems and Software, 65, 105-114.

6.2 Reverse engineering

Gail C. Murphy, David Notkin, and Kevin J. Sullivan. Software Reflexion Models: Bridging the Gap between Design and Implementation. IEEE Transactions on Software Engineering, 27 Bibliography 347(4):364–380, 2001.

Seonah Lee, Gail C. Murphy, Thomas Fritz, and Meghan Allen. How Can Diagramming Tools Help Support Programming Activities? In *VL/HCC'08*, pages 246–249, 2008.

Kenny Wong, Scott R. Tilley, Hausi A. M¨uller, and Margaret-Anne D. Storey. Structural Redocumentation: a Case Study. IEEE Software, 12(1):46–54, 1995.

Ducasse, S.; Lanza, M.; , "The class blueprint: visually supporting the understanding of

classes," *Software Engineering, IEEE Transactions on* , vol.31, no.1, pp. 75- 90, Jan. 2005

M.-A.D. Storey and H.A. Muller, “Manipulating and Documenting Software Structures Using Shrimp Views,” Proc. 1995 Int’l Conf. Software Maintenance, 1995.

M.-A. D. Storey, K. Wong, and H. A. Muller. 1997. How Do Program Understanding Tools Affect How Programmers Understand Programs. In Proceedings of the Fourth Working Conference on Reverse Engineering (WCRE '97) (WCRE '97).

S.G. Eick, J.L. Steffen, and S.E. Eric Jr., “SeeSoft—A Tool for Visualizing Line Oriented Software Statistics,” IEEE Trans. Software Eng., vol. 18, no. 11, pp. 957-968, Nov. 1992.

Chikofsky, E.J.; Cross, J.H., II; , "Reverse engineering and design recovery: a taxonomy," *Software, IEEE* , vol.7, no.1, pp.13-17, Jan 1990

D.J. Jerding, J.T. Stansko, and T. Ball, “Visualizing Interactions in Program Executions,” Proc. Int’l Conf. Software Eng. (ICSE ’97), pp. 360-370, 1997.

Colin J Bennett, Del Myers, Margaret-Anne Storey, Daniel M. German, David Ouellet, Martin Salois, and Philippe Charland. A survey and evaluation of tool features for understanding reverse-engineered sequence diagrams. J. Softw. Maint. Evol., 20(4):291–315, 2008.

W.J. Dzidek, E. Arisholm, and L.C. Briand. (2008). A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. IEEE Transactions on Software Engineering, 34 (3):407–432, May-June 2008.

6.3 Reading code (program comprehension), mental models of programs, and effects of expertise

(Edited and Presented by Stephen Oney -- see slides of overview presentation)

Primary:

Soloway, Elliot; Ehrlich, Kate; , "Empirical Studies of Programming Knowledge," *Software Engineering, IEEE Transactions on* , vol.SE-10, no.5, pp.595-609, Sept. 1984.

Storey, Margaret-Anne (2005) Theories, Methods, and Tools in Program Comprehension: Past, Present, and Future. Proceedings of the 13th International Workshop on Program Comprehension, pp. 181-191

Mayrhauser, A. and Vans, A. (1993) From Program Comprehension to Tool Requirements for an Industrial Environment. IEEE Workshop on Program Comprehension

Secondary:

Brooks, R. (1983) Towards a theory of the comprehension of computer programs. International Journal of Man-Machine Studies

Pennington, Nancy (1987) Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs. Cognitive Psychology, pp. 295-341, vol 19

Ben Liblit, Andrew Begel & Eve Sweetser. Cognitive Perspectives on the Role of Naming in Computer Programs [PDF]. PPIG06

CYNTHIA L. CORRITORE, SUSAN WIEDENBECK, An exploratory study of program comprehension strategies of procedural and object-oriented programmers, International Journal of Human-Computer Studies, Volume 54, Issue 1, January 2001, Pages 1-23,

Tertiary:

Steve McConnell. (2011). What does 10x mean? Measuring variations in programmer productivity. In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 567-573.

H. Sackman, W. J. Erikson, E. E. Grant. Exploratory experimental studies comparing online and offline programming performance. Commun. ACM, Vol. 11, No. 1. (January 1968), pp. 3-11.

B. Curtis. Substantiating programmer variability. Proceedings of the IEEE, Vol. 69, No. 7. (1981), pp. 846-846.

Dave Binkley, Dawn Lawrie, Steve Maex, Christopher Morrell, Identifier length and limited programmer memory, Science of Computer Programming, Volume 74, Issue 7, Special Issue on Program Comprehension (ICPC 2008), 1 May 2009, Pages 430-445

Françoise Détienne. 2001. Software Design---Cognitive Aspects. Frank Bott (Ed.). Springer-Verlag New York, Inc., New York, NY, USA.

Donald Knuth (1986) Literate Programming. The Computer Journal, Vol. 29, No. 3 pp. 201-211

6.4 Software Visualization

(Edited and Presented by Brad Myers -- see slides of overview presentation)

B.A. Price, R.M. Baecker, and I.S. Small (1993), "A Principled Taxonomy of Software Visualization," *J. Visual Languages and Computing*, vol. 4, no. 3, pp. 211-266, 1993.

J.I. Maletic, A. Marcus, and M. Collard, "A Task Oriented View of Software Visualization," Proc.

First Workshop Visualizing Software for Understanding and Analysis (VISSOFT '02), pp. 32-40, June 2002. pdf

Baecker, R., DiGiano, C., and Marcus, A., "Software Visualization for Debugging." *Communications of the ACM*, 1997. 40(4): pp. 44-54.

John Stasko, John Domingue, Marc H. Brown, and Blaine A. Price (Eds.). *Software Visualization: Programming as a Multimedia Experience*. 1998, The MIT Press. (Edited Book) (Google Books)

Brad A. Myers. "Visual Programming, Programming by Example, and Program Visualization; A Taxonomy," *Proceedings SIGCHI '86: Human Factors in Computing Systems*. Boston, MA. April 13-17, 1986. pp. 59-66. pdf

Teyseyre, A.R.; Campo, M.R.; , "An Overview of 3D Software Visualization," *IEEE Transactions on Visualization and Computer Graphics*, , vol.15, no.1, pp.87-105, Jan.-Feb. 2009. IEEE dl

6.4.1 Algorithm Visualization

Christopher D. Hundhausen, Sarah A. Douglas, and John T. Stasko. (2002). A meta-study of algorithm visualization effectiveness. In *Journal of Visual Languages and Computing*, 13, 259-290.

Gruia-Catalin Roman and Kenneth C. Cox. (1993). A taxonomy of program visualization systems. *IEEE Computer,* Volume 26 Issue 12, December 1993 . pp. 11-24. IEEE DL

J.T.Stasko & C.Patterson (1992). Understanding and characterizing software visualization systems. In Proceedings of the 1992 *IEEE Symposium on Visual Languages*, 15-18 Sep 1992, Seattle, WA, pp. 3-10. IEEE DL

Baecker, Ronald M. (1981). Sorting Out Sorting. narrated colour videotape, 30 minutes, presented at ACM SIGGRAPH '81 and excerpted in ACM SIGGRAPH Video Review #7, 1983. Los Altos, CA: Morgan Kaufmann. Google Video.

Brown, M. H. & Sedgewick, R. (1985). Techniques for Algorithm Animation. *IEEE Software*, 2(1): 28-39. IEEE pdf

Stasko, J. T. (1990). Tango: A Framework and System for Algorithm Animation. *IEEE Computer*, 23(9): 27-39. IEEE dl

Stasko, J., Badre, A., and Lewis, C. "Do Algorithm Animations Assist Learning? An Empirical

Study and Analysis," in *Proceedings INTERCHI'93: Human Factors in Computing Systems*. 1993. Amsterdam, The Netherlands: pp. 61-66. ACM dl

6.4.2 Code & Data Visualization

Myers, B.A. "Incense: A System for Displaying Data Structures," in *Proceedings SIGGRAPH'83; Computer Graphics*, 17. 1983. Detroit, MI. pp. 115-125. pdf

Baecker, R. and Marcus, A., "Design Principles for the Enhanced Presentation of Computer Program Source Text," in Proceedings of Chi'86 Conference on Human Factors in Computing Systems, M. Mantei and P. Orbeton, Editors. 1986, ACM. Boston. pp. 51-58. ACM dl

Steven P. Reiss, "PECAN: Program Development Systems that Support Multiple Views". IEEE Transactions on Software Engineering. SE-11(3). March, 1985. pp. 276 - 285. IEEE dl

Myers, B.A., Chandhok, R., and Sareen, A. "Automatic Data Visualization for Novice Pascal Programmers," in *IEEE Workshop on Visual Languages*. 1988. Pittsburgh, PA: pp. 192-198. pdf.

Reiss, S. P. (1990). Interacting with the FIELD Environment. *Software Practice and Experience*, 20(S-1): 89. (FIELD web site)

Eick, S.G. and Steffen, J.L. "Visualizing Code Profiling Line Oriented Statistics," in *IEEE Visualization*. 1992. Los Alamitos, CA: IEEE. ACM dl; pdf; video (10 min)

Abi-Antoun, M. and Aldrich, J. Static Extraction and Conformance Analysis of Hierarchical Runtime Architectural Structurusing Annotations. In *Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, 2009. [Paper (PDF)] [Slides (PDF)] [DOI] [Soundness proof is in Ph.D. thesis CMU-ISR-10-114]

7. Writing and Editing Code

7.1 Languages and tools for novice and end-user programmers

(To Be Edited and Presented by Brad Myers -- see slides of overview presentation)

Caitlin Kelleher and Randy Pausch. "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers " ACM Comput. Surv. 2005. 37(2). pp. 83-137. ACM

Guzdial, M. Programming environments for novices. In Computer Science Education Research, S. Fincher and M. Petre, Eds. Taylor & Francis, Abingdon, U.K., 2004, 127–154.

C. Kelleher, R. Pausch and S. Kiesler. "Storytelling Alice Motivates Middle School Girls to Learn Computer Programming," CHI'2007: Conference on Human Factors in Computing Systems, 2007. pp. 1455-1464. ACM

Paul Gross and Caitlin Kelleher. "Non-programmers identifying functionality in unfamiliar code: strategies and barriers," J. Vis. Lang. Comput. 2010. 21(5). pp. 263-276. 1860347.

Gross, Paul A., Micah S. Herstand, Jordana W. Hodges, and Caitlin L. Kelleher, A Code Reuse Interface for Non-Programmer Middle School Students, in Proceedings of IUI: International Conference on Intelligent User Interfaces. 2010, Hong Kong, China. p. 219-228.

J.F. Pane, B.A. Myers, and L.B. Miller, "Using HCI Techniques to Design a More Usable Programming System," *2002 IEEE Symposia on Human Centric Computing Languages and Environments (HCC'02)*. Arlington, VA, September 3-6, 2002. pp. 198-206. http://www.cs.cmu.edu/~pane/handsdesign.html

Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman and Yasmin Kafai. "Scratch: Programming for All," Comm. ACM. 2009. 52(11). pp. 60-67.

Christopher D. Hundhausen, Sean F. Farley, and Jonathan L. Brown. 2009. Can direct manipulation lower the barriers to computer programming and promote transfer of training?: An experimental study. ACM Trans. Comput.-Hum. Interact. 16, 3, Article 13 (September 2009), 40 pages.

Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. "Automation and Customization of Rendered Web Pages," in *ACM Conference on User Interface Software and Technology (UIST)*. 2005. Seattle, WA: pp. 163-172. pdf

7.2 Programming by example

(To Be Edited and Presented by Kerry Chang -- see slides of overview presentation)

Jiun-Hung Chen and Daniel S. Weld. 2008. Recovering from errors during programming by demonstration. In Proceedings of the 13th international conference on Intelligent user interfaces(IUI '08). ACM, New York, NY, USA, 159-168.

Brad A. Myers. "Demonstrational Interfaces: A Step Beyond Direct Manipulation," *IEEE Computer*. August, 1992. vol. 25, no. 8. pp. 61-73.

Richard G. McDaniel and Brad A. Myers, "Getting More Out Of Programming-By-Demonstration." *Proceedings CHI'99: Human Factors in Computing Systems*. Pittsburgh, PA, May 15-20, 1999. pp. 442-449. pdf and video

Henry Lieberman, Ed. *Your Wish is My Command*. San Francisco, Morgan Kaufmann. 2001.

Allen Cypher, Daniel C. Halbert, David Kurlander, Henry Lieberman, David Maulsby, Brad A. Myers and Alan Turransky, Eds. *Watch What I Do: Programming by Demonstration*. Cambridge, MA, MIT Press. 1993.

Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. "Sikuli: Using GUI Screenshots for Search and Automation." UIST 2009, pp. 183-192. [pdf]

7.3 Visual languages

(To Be Edited and Presented by Vishal Dwivedi -- see slides of overview presentation)

Taxonomies:

Margaret Burnett's Visual Language Research Bibliography

Brad A. Myers. "Taxonomies of Visual Programming and Program Visualization," *Journal of Visual Languages and Computing*. vol. 1, no. 1. March, 1990. pp. 97-123. *(A draft version is available in pdf format).*

Empirical Studies

Kirsten N. Whitley. (1997). Visual Programming Languages and the Empirical Evidence For and Against. J. Vis. Lang. Comput. 8(1): 109-142.

T. R. G. Green, M. Petre. When Visual Programs are Harder to Read than Textual Programs. In Human-Computer Interaction: Tasks and Organisation, Proceedings ECCE-6 (6th European Conference Cognitive Ergonomics) (1992).

Thomas R. G. Green, Marian Petre: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. J. Vis. Lang. Comput. 7(2): 131-174 (1996)

Other Papers

Daniel L. Moody: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Trans. Software Eng. 35(6): 756-779 (2009)

Margaret M. Burnett, "Visual Programming" In the Encyclopedia of Electrical and Electronics

Engineering (John G. Webster, ed.), 1999

Avraham Leff and James T. Rayfield. 2007. Webrb: evaluating a visual domain-specific language for building relational web-applications. In Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications (OOPSLA '07). ACM, New York, NY, USA, 281-300.

Jonathan Edwards. 2007. No ifs, ands, or buts: uncovering the simplicity of conditionals. In Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications (OOPSLA '07). ACM, New York, NY, USA, 639-658.

Seung Chan Slim Lim and Peter Lucas. 2006. JDA: a step towards large-scale reuse on the web. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (OOPSLA '06). ACM, New York, NY, USA, 586-601.

DANIEL D. HILS,"Visual Languages and Computing Survey: Data Flow Visual Programming Languages", Journal of Visual Languages and Computing (1992) 3, 69-101

Christopher D. Wickens, "Engineering Psychology and Human Performance", 3rd Edition
J. H. Larkin and H. A. Simon. Why a diagram is (sometimes) worth ten thousand words. Cognitive Science, 11:65-99, 1987.

Specific Visual Languages that are currently in use:

- Labview
- Lego Mindstorms NXT programming language
- Scratch at MIT
- Yahoo Pipes
- Adobe Flash Catalyst
- Microsoft SketchFlow
- Business Modelling language, BPMN
- Microsoft Visual Programming Language (VPL) "is an application development environment designed on a graphical dataflow-based programming model" for robotics.
- Microsoft Kodu - language for programming games, from MSR now a product:
- Stagecast Creator, formerly Apple's Cocoa
- Action(s), your Personal Automation Assistant
- Apple Automator
- Agentsheets
- CoScripter - IBM Research
- Clarity

7.4 Textual languages & editors

(To Be Edited and Presented by Stephen Oney -- see slides of overview presentation)

7.4.1 Impact of syntax on usability, form of the textual language

Greg Little and Robert C. Miller. 2007. Keyword programming in java. In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE '07). ACM, New York, NY, USA, 84-93.

Andrew D. Eisenberg and Gregor Kiczales. 2007. Expressive programs through presentation extension. In *Proceedings of the 6th international conference on Aspect-oriented software development* (AOSD '07). ACM, New York, NY, USA, 73-84.d

Samuel Davis, Gregor Kiczales. Registration-based language abstractions. OOPSLA 2010: 754-773.

7.4.2 Structured editors

Tim Teitelbaum and Thomas Reps. 1981. The Cornell program synthesizer: a syntax-directed programming environment. *Commun. ACM* 24, 9 (September 1981), 563-573.

A N Habermann and D Notkin. 1986. Gandalf: software development environments. *IEEE Trans. Softw. Eng.* 12, 12 (December 1986), 1117-1127.

Ko, A. J., Aung, H., and Myers, B. A. (2005). Design Requirements for More Flexible Structured Editors from a Study of Programmers' Text Editing. ACM Conference on Human Factors in Computing Systems (CHI), Portland OR.

R. Chandhok and et al. "Programming Environments based on structure editing: The Gnome approach," *Proceedings of the National Computer Conference (NCC'85),* AFIPS, 1985. IEEE and pdf

Andrew J. Ko and Brad A. Myers. "Citrus: A Toolkit for Simplifying the Creation of Structured Editors for Code and Data." *ACM Symposium on User Interface Software and Technology, UIST'05*, October 23-26, 2005, Seattle, WA. pp. 3-12. pdf or ACM ref video

Ko, A. J. and Myers, B. A. (2006). Barista: An Implementation Framework for Enabling New Tools, Interaction Techniques and Views for Code Editors (2006). ACM Conference on Human Factors in Computing Systems (CHI),

7.4.3 Structured programming

Edsger Dijkstra (March 1968). Go To Statement Considered Harmful. *Communications of the*

ACM 11 (3): 147–148.

Edsger W. Dijkstra. 1972. Chapter I: Notes on structured programming. In *Structured programming*, O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare (Eds.). Academic Press Ltd., London, UK, UK 1-82.

7.4.4 Object-oriented programming

Ignatios S. Deligiannis, Martin Shepperd, Steve Webster, and Manos Roumeliotis. 2002. A Review of Experimental Investigations into Object-Oriented Technology. Empirical Softw. Engg. 7, 3 (September 2002), 193-231.

John A. Lewis, Sallie M. Henry, Dennis G. Kafura, and Robert S. Schulman. 1991. An empirical study of the object-oriented paradigm and software reuse. In Conference proceedings on Object-oriented programming systems, languages, and applications (OOPSLA '91), Andreas Paepcke (Ed.). ACM, New York, NY, USA, 184-196.

Jinwoo Kim, F. Javier Lerch, and Herbert A. Simon. 1995. Internal representation and rule development in object-oriented design. ACM Trans. Comput.-Hum. Interact. 2, 4 (December 1995), 357-390.

Brad A. Myers, Richard G. McDaniel, Robert C. Miller, Alan Ferrency, Andrew Faulring, Bruce D. Kyle, Andrew Mickish, Alex Klimovitski, and Patrick Doane. "The Amulet Environment: New Models for Effective User Interface Software Development", *IEEE Transactions on Software Engineering*, Vol. 23, no. 6. June, 1997. pp. 347-365. IEEE pdf

Lutz Prechelt. (2011). Two comparisons of programming languages. In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 239-258.

7.4.5 Aspect-oriented programming

Terry Hon and Gregor Kiczales. 2006. Fluid AOP join point models. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*(OOPSLA '06). ACM, New York, NY, USA, 712-713. DOI=10.1145/1176617.1176687 http://doi.acm.org/10.1145/1176617.1176687

Adrian Colyer and Andrew Clement. 2004. Large-scale AOSD for middleware. In Proceedings of the 3rd international conference on Aspect-oriented software development (AOSD '04). ACM, New York, NY, USA, 56-65.

Robert J. Walker, Elisa L. A. Baniassad, and Gail C. Murphy. 1999. An initial assessment of aspect-oriented programming. In Proceedings of the 21st international conference on Software engineering (ICSE '99). ACM, New York, NY, USA, 120-130.

7.5 Software Evolution

(To Be Edited and Presented by YoungSeok Yoon -- see slides of overview presentation)

7.5.1 Code clone detection

Stefan Bellon, Rainer Koschke, Giulio Antoniol, Jens Krinke, Ettore Merlo, "Comparison and Evaluation of Clone Detection Tools," IEEE Transactions on Software Engineering, pp. 577-591, September, 2007.

Chanchal Kumar Roy and James R. Cordy, "A Survey on Software Clone Detection Research," SCHOOL OF COMPUTING TR 2007-541, QUEEN'S UNIVERSITY

Heejung Kim, Yungbum Jung, Sunghun Kim, and Kwangkeun Yi, "MeCC: Memory Comparison-based Clone Detector," In *Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011)*, Waikiki, Honolulu, Hawaii, May 21-28, 2011

7.5.2 Copy & paste

Miryung Kim, Lawrence Bergman, Tessa Lau, David Notkin, "An Ethnographic Study of Copy and Paste Programming Practices in OOPL," Empirical Software Engineering, International Symposium on, pp. 83-92, 2004 International Symposium on Empirical Software Engineering (ISESE'04), 2004

C. Kapser and M. W. Godfrey, "'Cloning considered harmful' considered harmful: Patterns of cloning in software," Empir.Softw. Eng., vol. 13, no. 6, pp. 645–692, 2008.

Toomim, M.; Begel, A.; Graham, S.L.; , "Managing Duplicated Code with Linked Editing," *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on* , vol., no., pp.173-180, 30-30 Sept. 2004.

Jonathan Edwards. 2005. Subtext: uncovering the simplicity of programming. In Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '05). ACM, New York, NY, USA, 505-518.

Miryung Kim, Vibha Sazawal, David Notkin, Gail Murphy. An empirical study of code clone genealogies. September 2005. ESEC/FSE-13.

Suresh Thummalapenta ·Luigi Cerulo · Lerina Aversano ·Massimiliano Di Pent. An empirical study on the maintenance of source code clones. Empir Software Eng (2010) 15:1–34

Ekwa Duala-Ekoko and Martin P. Robillard. Tracking Code Clones in Evolving Software. In Proceedings of the 29th International Conference on Software Engineering, pages 158-167, May 2007.

7.5.3 Refactoring

Don Roberts, John Brant, and Ralph Johnson. A refactoring tool for Smalltalk. Theory and Practice of Object Systems, 3(4):253–263, 1997.

Zhenchang Xing, Eleni Stroulia. (2006). Refactoring practice: How it is and how it should be supported — an Eclipse case study. In ICSM ’06: Proceedings of the 22nd IEEE International Conference on Software Maintenance.

Emerson Murphy-Hill and Andrew P. Black. Breaking the Barriers to Successful Refactoring: Observations and Tools for Extract Method. International Conference on Software Engineering. 2008.

Emerson Murphy-Hill, Chris Parnin, Andrew P. Black. How we refactor, and how we know it. In ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering (2009), pp. 287-297.

Mika V. M¨antyl¨a. An experiment on subjective evolvability evaluation of object-oriented software: explaining factors and interrater agreement. In Proceedings of the International Symposium on Empirical Software Engineering, pages 287–296, November 2005. doi: 10.1109/ISESE.2005.1541837.

Marat Boshernitsan, Susan L. Graham, and Marti A. Hearst. Aligning development tools with the way programmers think about code changes. In CHI ’07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 567–576.

Miryung Kim, Dongxiang Cai and Sunghun Kim, “An Empirical Investigation into the Role of API-Level Refactorings during Software Evolution,” In *Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011)*, Waikiki, Honolulu, Hawaii, May 21-28, 2011

7.5.4 Reviewing changes

Miryung Kim and David Notkin. Discovering and Representing Systematic Code Changes. ICSE09, 309-319.

Z. Xing and E. Stroulia, "UMLDiff: an algorithm for object-oriented design differencing," Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, p. 54–65, 2005.

7.6 Navigating working sets

(To Be Edited and Presented by Brad Myers -- see slides of overview presentation)

7.6.1 Exploratory studies

Ko, A. J., Aung, H., and Myers, B. A. (2005). Eliciting Design Requirements for Maintenance-Oriented IDEs: A Detailed Study of Corrective and Perfective Maintenance Tasks. *International Conference on Software Engineering (ICSE'05).* St. Louis, MO: pp. 126-135. (Winner, Distinguished Paper Award). pdf

Robert DeLine, Amir Khella, Mary Czerwinski, and George Robertson, Towards understanding programs through wear-based filtering, in Proceedings of the ACM Symposium on Software Visualization, May 2005.

B de Alwis, GC Murphy (2006). Using Visual Momentum to Explain Disorientation in the Eclipse IDE. In Proceedings of the IEEE Symposium on Visual Languages and Human Centric Computing (VL/HCC), Brighton, UK.

7.6.2 Lists of methods

Peri Tarr, Harold Ossher, William Harrison, and Stanley M. Sutton, Jr.. 1999. N degrees of separation: multi-dimensional separation of concerns. In Proceedings of the 21st international conference on Software engineering (ICSE '99). ACM, New York, NY, USA, 107-119.

Janice Singer, Robert Elves, Margaret-Anne D. Storey: NavTracks: Supporting Navigation in Software Maintenance. ICSM 2005: 325-334.

Mik Kersten and Gail C. Murphy. 2006. Using task context to improve programmer productivity. In Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering (SIGSOFT '06/FSE-14). ACM, New York, NY, USA, 1-11.

Martin P. Robillard and Gail C. Murphy. Representing Concerns in Source Code. ACM Transactions on Software Engineering and Methodology, 16(1):1-38, February 2007.

Curtis Fraser, Chris Luce, Jamie Starke and Jonathan Sillito. Tool Support for Working with Sets of Source Code Entities. In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing, 2008.

7.6.3 Tagging

Margaret-Anne D. Storey, Jody Ryall, Janice Singer, Del Myers, Li-Te Cheng, Michael J. Muller: How Software Developers Use Tagging to Support Reminding and Refinding. IEEE Trans. Software Eng. 35(4): 470-483 (2009).

7.6.4 Method recommenders

DeLine, R.; Czerwinski, M.; Robertson, G.; , "Easing program comprehension by sharing navigation data," *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on* , vol., no., pp. 241- 248, 20-24 Sept. 2005

Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, Andreas Zeller: Mining Version Histories to Guide Software Changes. IEEE Trans. Software Eng. 31(6): 429-445 (2005)

7.6.5 Maps of code

Michael J. Coblenz, Andrew J. Ko and Brad A. Myers, "JASPER: An Eclipse Plug-In to Facilitate Software Maintenance Tasks", *Eclipse Technology eXchange (ETX) Workshop at OOPSLA 2006*, Portland, Oregon, October 22-23, 2006. pp. 65-69. pdf and ACM DOI

Vineet Sinha, David Karger, Rob Miller, "Relo: Helping Users Manage Context during Interactive Exploratory Visualization of Large Codebases", Visual Languages and Human-Centric Computting (VL/HCC 2006). Sep. 4-8, 2006.

Robert DeLine, Gina Venolia, and Kael Rowan, Software Development with Code Maps, in Communications of the ACM, vol. 53, no. 8, pp. 48-54, Association for Computing Machinery, Inc., 4 July 2010

Andrew Bragdon, Steven P. Reiss, Robert Zeleznik, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola Jr. Code Bubbles: Rethinking the User Interface Paradigm of Integrated Development Environments. Proceedings of the 32nd International Conference on Software Engineering (2010).

Andrew Bragdon, Robert Zeleznik, Steven P. Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola, Jr.. 2010. Code bubbles: a working set-based interface for code understanding and maintenance. In *Proceedings of the 28th international conference on Human factors in computing systems* (CHI '10). ACM, 2503-2512. http://doi.acm.org/10.1145/1753326.1753706

8. Bugs

8.1 Causes of bugs, preventing bugs

(To Be Edited and Presented by Joshua Sunshine -- see slides of overview presentation)

8.1.1 Causes of bugs

Nachiappan Nagappan, Thomas Ball, Andreas Zeller. “Mining Metrics to Predict Component Failures.” Proceedings of the International Conference on Software Engineering, Shanghai, China, May 2006. ACM DL

Dewayne Perry. (2011). Where do most software flaws come from? In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 453-494.

Knuth, D. E. (1989), The errors of tex. Software: Practice and Experience, 19: 607–685.

Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem, and Dawson Engler. (2001). An empirical study of operating systems errors. In Proceedings of the eighteenth ACM symposium on Operating systems principles (SOSP '01). ACM, New York, NY, USA, 73-88.

R. L. Glass. 1981. Persistent Software Errors. IEEE Trans. Softw. Eng. 7, 2 (March 1981), 162-168.

Ko, A. J. and Myers, B. A. (2005). A Framework and Methodology for Studying the Causes of Software Errors in Programming Systems. Journal of Visual Languages and Computing, 16, 1-2, 41-84.

Saff, D.; Ernst, M.D.; , "Reducing wasted development time via continuous testing," *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on* , vol., no., pp. 281-292, 17-20 Nov. 2003

Marc Eaddy, Thomas Zimmermann, Kaitlin D. Sherwood, Vibhav Garg, Gail C. Murphy, Nachiappan Nagappan, Alfred V. Aho. Do Crosscutting Concerns Cause Defects?. In IEEE Transactions on Software Engineering (34): 497-515 (2008), July 2008, pp. 497-515.

John C. Knight , Nancy G. Leveson. (1986). An Experimental Evaluation Of The Assumption Of Independence In Multi-Version Programming. In Transactions on Software Engineering.

8.1.2 Preventing bugs with static analysis, causes of these bugs

Ko, A. J. and Wobbrock, J.O. (2010). Cleanroom: Edit-Time Error Detection with the Uniqueness Heuristic. IEEE Symposium on Visual Languages and Human-Centric Computing, Madrid, Spain, September 22-24, 7-14.

Yichen Xie and Dawson Engler. (2003). Using redundancies to find errors. Transactions on Software Engineering (TSE), 29 (10), 915-928.

Ciera Jaspan, I-Chin Chen, and Anoop Sharma. Understanding the value of program analysis tools, in the Companion Proceedings OOPSLA, Montreal, Quebec, Canada, 2007.

J. Zheng, L. Williams, N. Nagappan, W. Snipes, J. P. Hudepohl, and M. A. Vouk. On the value of static analysis for fault detection in software. IEEE Transactions on Software Engineering, 32(4):240–253, 2006.

Prechelt, L., Tichy W.: A Controlled Experiment to Assess the Benefits of Procedure Argument Type Checking, IEEE Transactions on Software Engineering 24(4), 1998, S. 302-312.

Nathaniel Ayewah and William Pugh. 2008. A report on a survey and study of static analysis users. In Proceedings of the 2008 workshop on Defects in large software systems (DEFECTS '08). ACM, New York, NY, USA, 1-5.

Stefan Hanenberg. 2010. An experiment about static and dynamic type systems: doubts about the positive impact of static type systems on development time. In Proceedings of the ACM international conference on Object oriented programming systems languages and applications(OOPSLA '10). ACM, New York, NY, USA, 22-35.

8.2 Reporting and triaging bugs

(To Be Edited and Presented by YoungSeok Yoon -- see slides of overview presentation)

Christine A. Halverson, Jason B. Ellis, Catalina Danis, and Wendy A. Kellogg. 2006. Designing task visualizations to support the coordination of work in software development. In Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (CSCW '06). ACM, New York, NY, USA, 39-48.

Jason B. Ellis, Shahtab Wahid, Catalina Danis, and Wendy A. Kellogg. 2007. Task and social visualization in software development: evaluation of a prototype. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '07). ACM, New York, NY, USA, 577-586.

Ko, A. J. and Chilana, P.K. (2011). Design, Discussion, and Dissent in Open Bug Reports. iConference, Seattle, WA, February 8-11,

Ko, A. J. and Chilana P. (2010). How Power Users Help and Hinder Open Bug Reporting. ACM Conference on Human Factors in Computing Systems (CHI), Atlanta, GA, 1665-1674.

Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, Brendan Murphy. Characterizing and Predicting Which Bugs Get Fixed: An Empirical Study of Microsoft Windows. In Proceedings of the 32th International Conference on Software Engineering (ICSE 2010), Cape Town, South Africa, May 2010.

Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, Thomas Zimmermann. What Makes a Good Bug Report?. In Proceedings of the 16th International Symposium on Foundations of Software Engineering (FSE 2008), Atlanta, GA, USA, November 2008.

Philip J. Guo, Dawson Engler. Linux Kernel Developer Responses to Static Analysis Bug

Reports (short paper). In Proceedings of the 2009 USENIX Annual Technical Conference, June 2009.

Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, Brendan Murphy. "Not My Bug!" and Other Reasons for Software Bug Report Reassignments. In Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 2011), Hangzhou, China, March 2011.

Dane Bertram, Amy Voida, Saul Greenberg, Robert Walker. Communication, Collaboration, and Bugs: The Social Nature of Issue Tracking in Small, Collocated Teams. CSCW 2010.

Barcellini, F., Detienne, F., Burkhardt, J.M., Sack, W. (2008). A socio-cognitive analysis of online design discussions in an open source software community. Interacting with Computers, 20, 141-165.

Anvik, J., Hiew, L., and Murphy, G.C. (2006). Who should fix this bug? Proc. ICSE 2006, ACM Press (2006), 361–370.

Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann. 2009. Improving bug triage with bug tossing graphs. In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC/FSE '09). ACM, New York, NY, USA, 111-120.

Twidale, M.B.; Nichols, D.M.; , "Exploring Usability Discussions in Open Source Development," *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on* , vol., no., pp. 198c, 03-06 Jan. 2005

Ko, A. J., Myers, B.A., Chau, D. H. (2006). A Linguistic Analysis of How People Describe Software Problems in Bug Reports. Visual Languages and Human-Centric Computing.

8.3 Debugging

(Edited and Presented by Thomas LaToza -- see slides of overview presentation)

8.3.1 Strategies

Mark Weiser. 1982. Programmers use slices when debugging. Commun. ACM 25, 7 (July 1982), 446-452.

Katz, I. R. & Anderson, J. R. (1987). Debugging: An Analysis of Bug-Location Strategies. Human–Computer Interaction, 3(4), 351-399.

D.J. Gilmore. (1991). Models of debugging. *Acta Psychologica, 78*, 151-172.

Eisenstadt, M. Tales of Debugging from the Front Lines. Proc. Empirical Studies of Programmers, Ablex Publishing, Norwood, NJ, 1993, 86-112.

Francel M. A. and S. Rugaber (2001). The Value of Slicing While Debugging, *Science of Computer Programming*, 40(2-3), 151-169.

Pablo Romero, Benedict du Boulay, Richard Cox, Rudi Lutz, and Sallyann Bryant. 2007. Debugging strategies and tactics in a multi-representation software environment. Int. J. Hum.-Comput. Stud.65, 12 (December 2007), 992-1009.

Scott D. Fleming, Eileen Kraemer, R. E. K. Stirewalt, Shaohua Xie, and Laura K. Dillon. 2008. A study of student strategies for the corrective maintenance of concurrent software. In Proceedings of the 30th international conference on Software engineering (ICSE '08).

Joseph Lawrance, Christopher Bogart, Margaret Burnett, Rachel Bellamy, Kyle Rector, Scott D. Fleming, How Programmers Debug, Revisited: An Information Foraging Theory Perspective, IEEE Transactions on Software Engineering (to appear).

8.3.2 Tools

Andrew J. Ko. Asking and Answering Questions about the Causes of Software Behavior. Dissertation, Carnegie Mellon University. See only Section 2.3. "Program understanding tools", p24-33 for a review of debugging tools.

M. V. Zelkowitz. 1973. Reversible execution. Commun. ACM 16, 9 (September 1973), 566-.

Henry Lieberman and Christopher Fry. 1995. Bridging the gulf between code and behavior in programming. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '95), 480-486.

Bill Lewis. Debugging backwards in time. In Proceedings of the Fifth International Workshop on Automated Debugging (AADEBUG 2003), October 2003.

Andrew J. Ko and Brad A. Myers. 2010. Extracting and answering why and why not questions about Java program output. ACM Trans. Softw. Eng. Methodol. 20, 2, Article 4 (September 2010), 36 pages.

Ko, A. J. and Myers B.A. (2009). Finding Causes of Program Output with the Java Whyline. ACM Conference on Human Factors in Computing Systems (CHI '09), Boston, MA, 1569-1578.

Bjoern Hartmann, Daniel MacDougall, Joel Brandt, Scott R. Klemmer. What Would Other Programmers Do? Suggesting Solutions to Error Messages. CHI: ACM Conference on Human Factors in Computing Systems, Atlanta, GA, 2010.

Andreas Zeller and Ralf Hildebrandt. Simplifying and Isolating Failure-Inducing Input. IEEE Transactions on Software Engineering 28(2), February 2002, pp. 183-200.

Hiralal Agrawal, Richard A. Demillo, and Eugene H. Spafford. 1993. Debugging with dynamic slicing and backtracking. Softw. Pract. Exper. 23, 6 (June 1993), 589-616.

Andreas Zeller. Automated Debugging: Are We Close? IEEE Computer, November 2001.

Ben Liblit. (2005). Cooperative bug isolation. Dissertation, UC Berkeley.

Ben Liblit, Mayur Naik, Alice X. Zheng, Alex Aiken, and Michael I. Jordan. 2005. Scalable statistical bug isolation. In Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation (PLDI '05). ACM, New York, NY, USA, 15-26.

James A. Jones and Mary Jean Harrold. 2005. Empirical evaluation of the tarantula automatic fault-localization technique. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering (ASE '05). ACM, New York, NY, USA, 273-282.

Wim De Pauw and Gary Sevitsky. Visualizing Reference Patterns for Solving Memory Leaks in Java. In European Conference on Object-Oriented Programming (ECOOP), pages 116–134, 1999.

9. End-User Software Engineering

9.1 End-User Software Engineering in General

(To Be Edited and Presented by Vishal Dwivedi -- see slides of overview presentation)

9.1.1 Surveys

Ko, A. J., Abraham R., Beckwith L., Blackwell A., Burnett M.M., Erwig M., Scaffidi C., Lawrence J., Lieberman H., Myers B.A., Rosson M.B., Rothermel G., Shaw M. and Wiedenbeck S. (in press). The State of the Art in End-User Software Engineering, ACM Computing Surveys, to appear.

9.1.2 Exploratory Studies

Christopher Scaffidi, Mary Shaw, Brad A. Myers: Estimating the Numbers of End Users and End User Programmers. VL/HCC 2005: 207-214. pdf

Karen Rothermel, Curtis Cook, Margaret Burnett, Justin Schonfeld, Thomas Green, and Gregg Rothermel, "WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation", International Conference on Software Engineering, Limerick, Ireland, 230-239, June 2000.pdf

Stephen G. Powell, Kenneth R. Baker, Barry Lawson (2007-12-01). "A Critical Review of the Literature on Spreadsheet Errors", Decision Support Systems 46 (2008) 128–138. pdf

Margaret M. Burnett, Scott D. Fleming, Shamsi Iqbal, Gina Venolia, Vidya Rajaram, Umer Farooq, Valentina Grigoreanu, Mary Czerwinski: Gender differences and programming environments: across programming populations. ESEM 2010. pdf

Ruthruff J., Burnett M., and Rothermel G. 2005. An empirical study of fault localization for end-user programmers. International Conference on Software Engineering, St. Louis, Missouri, May, 352-361. pdf

9.1.3 Tools and Approaches

Margaret M. Burnett: What Is End-User Software Engineering and Why Does It Matter? IS-EUD 2009: 15-28 (pdf)

Chris Scaffidi, Brad Myers, Mary Shaw. "Intelligently Creating and Recommending Reusable Reformatting Rules". *IUI'2009: Intelligent User Interfaces Conference*, Sanibel Island, Florida, 8-11 February 2009. pp. 297-306. pdf

Christopher Scaffidi, Brad Myers, Mary Shaw, "Topes: Reusable Abstractions for Validating Data." *ICSE'2008: 30th International Conference on Software Engineering*, Leipzig, Germany, 10 - 18 May 2008. pp. 1-10. IEEE DL pdf

Sunyoung Park, Brad Myers, Andrew Ko. "Designers' Natural Descriptions of Interactive Behaviors," *2008 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC'08*. Sept 15-18, 2008, Herrsching am Ammersee, Germany. pp. 185-188. pdf

Jeffrey Wong and Jason I. Hong. 2007. Making mashups with marmite: towards end-user programming for the web. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '07). ACM, New York, NY, USA, 1435-1444.

Lin, James, Jeffrey Wong, Jeffrey Nichols, Allen Cypher, and Tessa A. Lau, End-User Programming of Mashups with Vegemite, in Proceedings of IUI: International Conference on Intelligent User Interfaces. 2009, Sanibel Island, Florida. p. 97-106.

Little, Greg, Tessa A. Lau, Allen Cypher, James Lin, Eben M. Haber, and Eser Kandogan, Koala: Capture, Share, Automate, Personalize Business Processes on the Web, in Proceedings of CHI: ACM Conference on Human Factors in Computing Systems. 2007, San Jose, California.

p. 943-946.

Umarji, M., Pohl, M., Seaman, C., Koru, A. G., and Liu, H. 2008. Teaching software engineering to end-users. International Workshop on End-User Software Engineering, Leipzig, Germany, May 40-42.

9.2 End-User Software Engineering for Scientists

(To Be Edited and Presented by Cyrus Omar -- see slides of overview presentation)

9.2.1 Exploratory studies

Basili, V.R.; Carver, J.C.; Cruzes, D.; Hochstein, L.M.; Hollingsworth, J.K.; Shull, F.; Zelkowitz, M.V.; , "Understanding the High-Performance-Computing Community: A Software Engineer's Perspective," *Software, IEEE* , vol.25, no.4, pp.29-36, July-Aug. 2008

Luke Nguyen-Hoan, Shayne Flint, and Ramesh Sankaranarayana. 2010. A survey of scientific software development. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10). ACM, New York, NY, USA, , Article 12 , 10 pages.

Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post. 2007. Software Development Environments for Scientific and Engineering Software: A Series of Case Studies. In Proceedings of the 29th international conference on Software Engineering (ICSE '07). IEEE Computer Society, Washington, DC, USA, 550-559.

Judith Segal. 2007. Some Problems of Professional End User Developers. In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC '07). IEEE Computer Society, Washington, DC, USA, 111-118.

Judith Segal. 2008. Models of Scientific Software Development. *In Proc. 2008 Workshop Software Eng. in Computational Science and Eng. (SECSE08)*. 13 May 2008, Leipzig, Germany.

Judith Segal. 2009. Software Development Cultures and Cooperation Problems: A Field Study of the Early Stages of Development of Software for a Scientific Community. Comput. Supported Coop. Work 18, 5-6 (December 2009), 581-606.

Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. 2009. How do scientists develop and use scientific software?. In Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering (SECSE '09). IEEE Computer Society, Washington, DC, USA, 1-8.

Rebecca Sanders and Diane Kelly. 2008. Dealing with Risk in Scientific Software Development. IEEE Softw. 25, 4 (July 2008), 21-28.

Les Hatton and Andy Roberts. 1994. How Accurate is Scientific Software? IEEE Trans. Softw. Eng. 20, 10 (October 1994), 785-797.

Daniel Hook and Diane Kelly. (2009). Testing for trustworthiness in scientific software. In Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering (SECSE '09). IEEE Computer Society, Washington, DC, USA, 59-64.

James Howison and Jim Herbsleb. (2011). "Scientific software production: incentives and collaboration". CSCW 2011.

9.2.2 Languages and tools

Letondal, C. 2006. Participatory Programming: Developing Programmable Bioinformatics Tools for End--Users, in Lieberman, H., Paternò, F., and Wulf, V. (Eds) 2006. End User Development. Springer, Dordrecht, The Netherlands., 207--242.

J.P. Massar, Michael Travers, Jeff Elhai, and Jeff Shrager. (2005). BioLingua: a programmable knowledge environment for biologists. In Bioinformatics, 21(1), 199-207.

Elhai J, Taton A, Massar JP, Myers JK, Travers M, Casey J, Slupesky M, Shrager J.. BioBIKE: A Web-based, programmable, integrated biological knowledge base. Nucleic Acids Res. 2009 Jul 1;37(Web Server issue):W28-32. Epub 2009 May 11.

Michael Pedersen and Andrew Phillips, Towards programming languages for genetic engineering of living cells, in Journal of the Royal Society Interface, 15 April 2009.

Aneil Mallavarapu, Matthew Thomson, Benjamin Ullian and Jeremy Gunawardena. Programming with models: modularity and abstraction provide powerful capabilities for systems biology. J. R. Soc. Interface 2009 6, 257-270.

Vigder, M.; Vinson, N.G.; Singer, J.; Stewart, D.; Mews, K.; , "Supporting Scientists' Everyday Work: Automating Scientific Workflows,"*Software, IEEE* , vol.25, no.4, pp.52-58, July-Aug. 2008.

10. Software Development Processes

(To Be Edited and Presented by Kun Niu -- see slides of overview presentation)

10.1 Pair programming

Laurie Williams. (2011). Pair programming. In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 311-328.

Andrew Begel, Nachiappan Nagappan. Pair Programming: What's in it for me?. In the 2nd

International Symposium on Empirical Software Engineering and Measurement (ESEM), Kaiserslautern, Germany. October 2008.

Jan Chong and Tom Hurlbutt. 2007. The Social Dynamics of Pair Programming. In Proceedings of the 29th international conference on Software Engineering (ICSE '07). IEEE Computer Society, Washington, DC, USA, 354-363.

10.2 Agile

Andrew Begel and Nachiappan Nagappan. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. In the First International Symposium on Empirical Software Engineering and Metrics (ESEM), Madrid, Spain, September 2007. pp. 255 - 264.

Teasley, S.D.; Covi, L.A.; Krishnan, M.S.; Olson, J.S.; , "Rapid software development through team collocation," *Software Engineering, IEEE Transactions on* , vol.28, no.7, pp. 671- 683, Jul 2002

Caryna Pinheiro, Frank Maurer and Jonathan Sillito. Adopting iterative development: the perceived business value. In Proceedings of the International Conference on Agile Processes and eXtreme Programming in Software Engineering, 2008.

Marisa Leavitt Cohn, Susan Elliott Sim, and Charlotte P. Lee. 2009. What Counts as Software Process? Negotiating the Boundary of Software Work Through Artifacts and Conversation.Comput. Supported Coop. Work 18, 5-6 (December 2009), 401-443.

Rashina Hoda, Philippe Kruchten, James Noble, and Stuart Marshall. 2010. Agility in context. In Proceedings of the ACM international conference on Object oriented programming systems languages and applications (OOPSLA '10). ACM, New York, NY, USA, 74-88.

Jenny Quillien, Dave West. 2010. Rubber ducks, nightmares, and unsaturated predicates: proto-scientific schemata are good for agile. In Proceedings of the ACM international conference on Object oriented programming systems languages and applications(OOPSLA '10). ACM, New York, NY, USA

10.3 Test-driven development

Hakan Erdogmus, Maurizio Morisio, and Marco Torchiano. 2005. On the Effectiveness of the Test-First Approach to Programming. IEEE Trans. Softw. Eng. 31, 3 (March 2005), 226-237.

Chetan Desai, David Janzen, and Kyle Savage. 2008 A survey of evidence for test-driven development in academia Bulletin. Volume 40 Issue 2, June 2008.

Thirumalesh Bhat, Nachiappan Nagappan. 2006 Evaluating the efficacy of test-driven

development: industrial case studies ISESE '06 Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering

Burak Turhan, Lucas Layman, Madeline Diep, Hakan Erdogmus, and Forest Shull. (2011). How effective is test-driven development? In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 207-217.

David Janzen, Hossein Saiedian. 2008 Test-driven learning in early programming courses SIGCSE '08 Proceedings of the 39th SIGCSE technical symposium on Computer science education, ACM New York, NY, USA ©2008

10.4 Code organization techniques

Walter Tichy. The evidence for design patterns. (2011). In Making Software: What really works and why we believe it, Andy Oram and Greg Wilson (eds), 393-414.

Stuart H. Zweben, Stephen H. Edwards, Bruce W. Weide, and Joseph E. Hollingsworth. 1995. The Effects of Layering and Encapsulation on Software Development Cost and Quality. *IEEE Trans. Softw. Eng*. 21, 3 (March 1995), 200-208.

10.5 Capability Maturity Model Integration (CMMI)

Presented by Prof. Mark Paulk -- see slides of overview presentation

Main CMMI Web Site

Mike Phillips and Sandy Shrum, “Process Improvement for All: What to Expect from CMMIVersion 1.3”, IEEE Crosstalk, January/February 2010, pp. 10-14, IEEE pdf or local pdf

MARK C. PAULK, BILL CURTIS, MARY BETH CHRISSIS, and CHARLES V. WEBER, "Capability Maturity Model, Version 1.1", *IEEE Software*. 10(4), Jul, 1993, pp. 18-27. IEEE DL or local pdf.

See also papers on Mark Paulk’s web page: http://www.cs.cmu.edu/~mcp/papers/mcppapers.html

11. Other topics

Motivations of Software Engineers

Tracy Hall, Nathan Baddoo, Sarah Beecham, Hugh Robinson, and Helen Sharp. 2009. A systematic review of theory use in studies investigating the motivations of software engineers. *ACM Trans. Softw. Eng. Methodol*. 18, 3, Article 10 (June 2009), 29 pages.

Open Source Software

Crowston, K., Wei, K, Howison, J., and Wiggins, A. (2011). Free/Libre Open Source Software Development: What We Know and What We Do Not Know. ACM Computing Surveys. Forthcoming.