Seminar
Requirements Engineering in Software Evolution

Prof. Martin Glinz
Emitzá Guzmán, Eya Ben Charrada

University of Zurich[UZH]

Requirements Engineering Research Group

Goal

Analyze and present scientific work in the field of requirements engineering

Organization

3 ECTS

Seminar Process

1. Write seminar paper (15 pages Springer style)
2. Peer-review (2 papers)
3. Present (20 min)
4. Participate in other presentations

Grading

2/3 seminar paper + reviews, 1/3 presentation + participation

Plan

Day	Activity
20.02	First meeting
24.02	Choosing topic deadline
~15.03, by appointment or email	Q&A seminar topic, paper structure, main storyline (voluntary)
~30.03, by appointment or email	Review and discussion of draft (voluntary)
20.04, 23:59 CET	Paper submission deadline
2.05, 23.59 CET	Peer-review submission deadline
16.05, 23:59 CET	Camera-ready submission deadline
~23.05, by appointment or email	Review and discussion of presentation (voluntary)
01.06	Final presentations

Q&A Seminar Topic

By appointment

Prepare the meeting

- Draft an outline
- For each planned section have a short description of what you plan to write

Send draft outline at least 2 days before the meeting

Review and Discussion of Draft

Read the material on writing before starting to write

Send draft 7 days *before* your appointment

Have a coherent version available (details may still be missing)

Peer Reviews

Review papers from two other students

Adhere to the reviewer forms (will be provided)

Be constructive, make concrete suggestions for improvement

Check for plagiarism, let us know if plagiarism is found!

Final Presentation Day

20 min. presentation + 10 min. Q&A and discussion

Each reviewer presents two questions in the discussion

Encouraged to participate in discussions where you are not reviewer!

Final Presentation Day

Participation on presentation day compulsory

Prepare presentation well, will be part of your grade!

Train your presentation with a peer beforehand

News and Information

Visit

http://www.ifi.uzh.ch/rerg/courses/fs17/sem-re-bsc.html

http://www.ifi.uzh.ch/rerg/courses/fs17/sem-re-msc.html

Seminar Theme

Requirements Engineering in Software Evolution

Main Themes

1. Elicitation and Automatic Analysis

1. Decision Making

1. Traceability and Management

Requirements Elicitation

- Which methods exist for eliciting requirements during software evolution?
- What are their advantages and limitations?
- **Focus on:** all methods except those that use social media/app stores OR only on app store/social media

References:
[1] Norbert Seyff, Florian Graf, Neil A. M. Maiden. Using Mobile RE Tools to Give End-Users Their Own Voice. RE 2010
[2] Kurt Schneider. Focusing spontaneous feedback to support system evolution. RE 2011
[3] Dennis Pagano, Walid Maalej: User feedback in the appstore: An empirical study. RE 2013
[4] Chen, N., Lin, J., Hoi, S. C., Xiao, X., & Zhang, B. (2014, May). AR-miner: mining informative reviews for developers from mobile app marketplace. ICSE 2014

Requirements Prioritization

- Which methods exist for prioritizing requirements during software evolution?
- What are their advantages and limitations?
- **Focus on:** Agile projects OR other SE methodologies

References:
[1] Berander, Patrik, and Anneliese Andrews. "Requirements prioritization." Engineering and managing software requirements. Springer Berlin Heidelberg, 2005. 69-94.
[2] Achimugu, Philip, et al. "A systematic literature review of software requirements prioritization research." Information and software technology 56.6 (2014): 568-585.
[3] Laura Lehtola, Marjo Kauppinen. "Empirical evaluation of two requirements prioritization methods in product development projects." European Conference on Software Process Improvement. Springer Berlin Heidelberg, 2004.

Release Planning

- Which release planning methods exist?
- In which context could these methods be applied?
- What are the limitations of these methods?
- **Focus on:** Agile projects OR other SE methodologies

References:
[1] Des Greer and Günther Ruhe. Software release planning: an evolutionary and iterative approach. Information and Software Technology 46.4 (2004): 243-253.
[2] Carlshamre Pär, and Björn Regnell. Requirements lifecycle management and release planning in market-driven requirements engineering processes. International Workshop on Database and Expert Systems Applications, 2000.
[3] Svahnberg, Mikael, et al. "A systematic review on strategic release planning models." Information and software technology 52.3 (2010): 237-248.

Requirements Traceability for Managing Software Evolution

- What is traceability?
- What the benefits of traceability and what does it cost?
- How can traces be established and maintained?
- How can traceability support the evolution of software systems?

References:

1] Gotel, Orlena, Jane Cleland-Huang, Jane Huffman Hayes, Andrea Zisman, Alexander Egyed, Paul Grünbacher, Alex Dekhtyar, Giuliano Antoniol, Jonathan Maletic, and Patrick Mäder. "Traceability fundamentals." In Software and Systems Traceability, pp. 3-22. Springer London, 2012.

[2] Ingram, C., and Riddle, S. (2012). Cost---benefits of traceability. In J. Cleland---Huang, O. Gotel, A. Zisman (eds.): Software and Systems Traceability (pp. 23---42). Springer London.

[3] Cleland-Huang, J., Settimi, R., Romanova, E., Berenbach, B., Clark, S., (2007). Best practices for automated traceability. IEEE Computer 40(6): 27-35.

[4] Rempel, P., and Mäder, P. (2015). Estimating the Implementation Risk of Requirements in Agile Software Development Projects with Traceability Metrics. In Requirements Engineering: Foundation for Software Quality(REFSQ 2015) (pp. 81-97). Springer.

[5] Cleland-Huang, J. (2012). Traceability in agile projects. In J. Cleland-Huang, O. Gotel, A. Zisman (eds.): Software and Systems Traceability (pp. 265-275). Springer London.

Co-evolving Software Artifacts

- When software evolve, are the requirements and other software artifacts kept up-to-date in practice? If not why?
- What approaches are there to support the update of software artifacts?

References:
[1] T. C. Lethbridge, J. Singer, and A. Forward, “How software engineers use documentation: The state of the practice,” IEEE Software, vol. 20, no. 6, pp. 35–39, 2003.
[2] Ben Charrada, Eya, Anne Koziolek, and Martin Glinz. "Supporting requirements update during software evolution." Journal of Software: Evolution and Process 27.3 (2015): 166-194.
[3] Mens, Kim, et al. "Co-evolving code and design with intensional views: A case study.” Computer Languages, Systems & Structures 32.2 (2006): 140-156.

Classifying Requirements Change

- What are the types of software and requirements change?
- What are the main causes and triggers for changes in requirements?
- What are the costs and effects of those changes?

References:

1] Buckley, Jim, et al. "Towards a taxonomy of software change." Journal of Software Maintenance and Evolution: Research and Practice 17.5 (2005): 309-332.

[2] McGee, Sharon, and Des Greer. "Towards an understanding of the causes and effects of software requirements change: two case studies." Requirements Engineering 17.2 (2012): 133-155.

[3] Nurmuliani, Nur, Didar Zowghi, and Susan P. Williams. "Characterising Requirements Volatility: An Empirical Case Study." ISESE. 2005.

Next Steps

Let us your topic preference after class

or

Send your three most favorite topics to guzman@ifi.uzh.ch, topics will be assigned on a first come, first serve basis.

Explicit User Feedback Analysis

- What challenges arise when analyzing explicit user feedback?
- What are current strategies for solving these issues?
- What are the limitations of these strategies?
- How can explicit feedback be useful for software evolution and requirements engineering?

References:
[1] Dennis Pagano, and Bernd Bruegge. User involvement in software evolution practice: a case study. ICSE 2013.
[2] Ning Chen, et al. AR-Miner: mining informative reviews for developers from mobile app marketplace. ICSE 2014.
[3] Gu, Xiaodong, and Sunghun Kim. What Parts of Your Apps are Loved by Users? ASE 2015.

Implicit User Feedback Analysis

- What challenges arise when analyzing implicit user feedback?
- What are current strategies for solving these issues?
- What are the limitations of these strategies?
- How can implicit feedback be useful for software evolution and requirements engineering?

References:
[1] Jansen, Bernard J. Understanding user-web interactions via web analytics. Synthesis Lectures on Information Concepts, Retrieval, and Services 1.1 (2009): 1-102.
[2] Nasraoui, Olfa, et al. A web usage mining framework for mining evolving user profiles in dynamic web sites. Transactions on Knowledge and Data Engineering, IEEE Transactions 20.2 (2008): 202-215.
[3] Benevenuto, Fabrício, et al. Characterizing user behavior in online social networks. IMC 2009.