A Game for Taking Requirements Engineering More Seriously

Presented at MERE'08, Barcelona

Eric Knauss, Kurt Schneider, Kai Stapel
{eric.knauss, kurt.schneider, kai.stapel}@inf.uni-hannover.de
Software Engineering Group, Leibniz Universität Hannover

9. September 2008

Motivation

- Requirements are often not taken seriously
  - Successful projects without (explicit) RE
  - Customer pays for the code, not for the RE efforts
  - Failed projects even with RE!
- Requirements are hard to teach
  - Hard to explain, where the problems come from

Example

- (Computer Science) Student has excellent technical knowledge
- Based on his Master Thesis he starts a Spin-Off and brings this technique into industry
- Acquisition of new Customers is focused around how his key technique can be leveraged
- Does not seem to be important, what the customer (really) wants

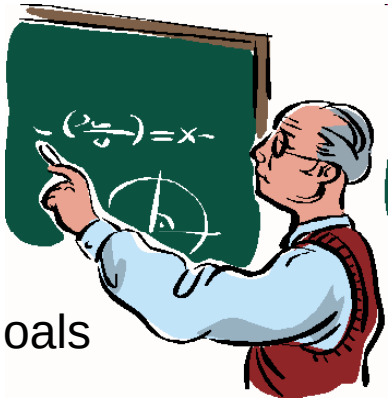
Overview

1. Related Work

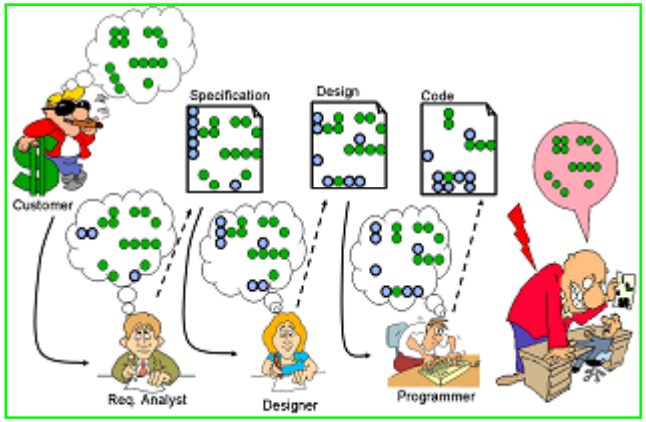
8. **References**

1. Abdel-Hamid, T. and S.E. Madnick, *Software Project Dynamics*. 1991, Englewood Cliffs, NJ: Prentice Hall.
2. Beck, K., *Extreme Programming Explained*. 2000: Addison-Wesley.
3. Beedle, M. and K. Schwaber, *Agile Software Development with Scrum*. 2001: Prentice Hall.
4. Deininger, M. and K. Schneider, *Teaching Software Project Management by Simulation - Experiences with a Comprehensive Model*. *Conference on Software Engineering Education (CSEE)*, 1994. Austin, Texas.
5. Forrester, J.W., *Industrial dynamics. 9. Edition*. 1977: Cambridge.
6. IABG. *V Model*, www.v-modell.iabg.de
7. Knauss, E., et al., *Nicht perfekt, aber richtig - Erfahrungen mit Software-Anforderungen*. Softwaretechnik Trends, 2007. 27(1).
8. Nonaka, I. and H. Takeuchi, *The Knowledge-Creating Company*. 17 ed. 1995: Oxford University Press.
9. Paulk, M.C., et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*. 1 ed. SEI Series in Software Engineering, ed. M.C. Paulk, et al. Vol. 1. 1994. Reading, MA: Addison Wesley Longman, Inc.
10. Rodriguez, D., M. Satpathy, and D. Pfahl, *Effective software project management education through simulation models. An externally replicated experiment*. *Conf. on Product Focused Software Process Improvement (PROFES)*, 2004. Kansai Science City, Japan: Bomarius, F.
11. Schneider, K. *A Descriptive Model of Software Development to Guide Process Improvement*. *Conquest*, 2004. Nürnberg, Germany: ASQF.
12. Schneider, K. *Generating Fast Feedback in Requirements Elicitation*. Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ 2007). Trondheim, Norway.
13. Schneider, K. and K. Nakakoji, *Collaborative Learning as Interplay between Simulation Model Builder and Player*. *Computer Supported Cooperative Learning (CSCL)*, 1995. Indianapolis: IN.

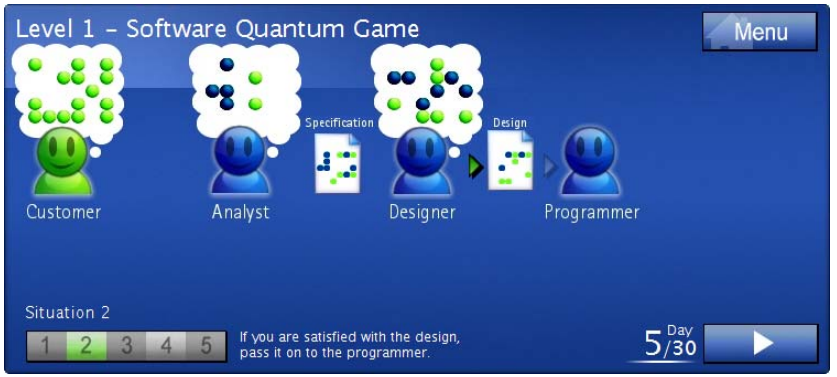
2. Goals



3. Basic Concepts



4. A Game for Learning



5. Conclusion

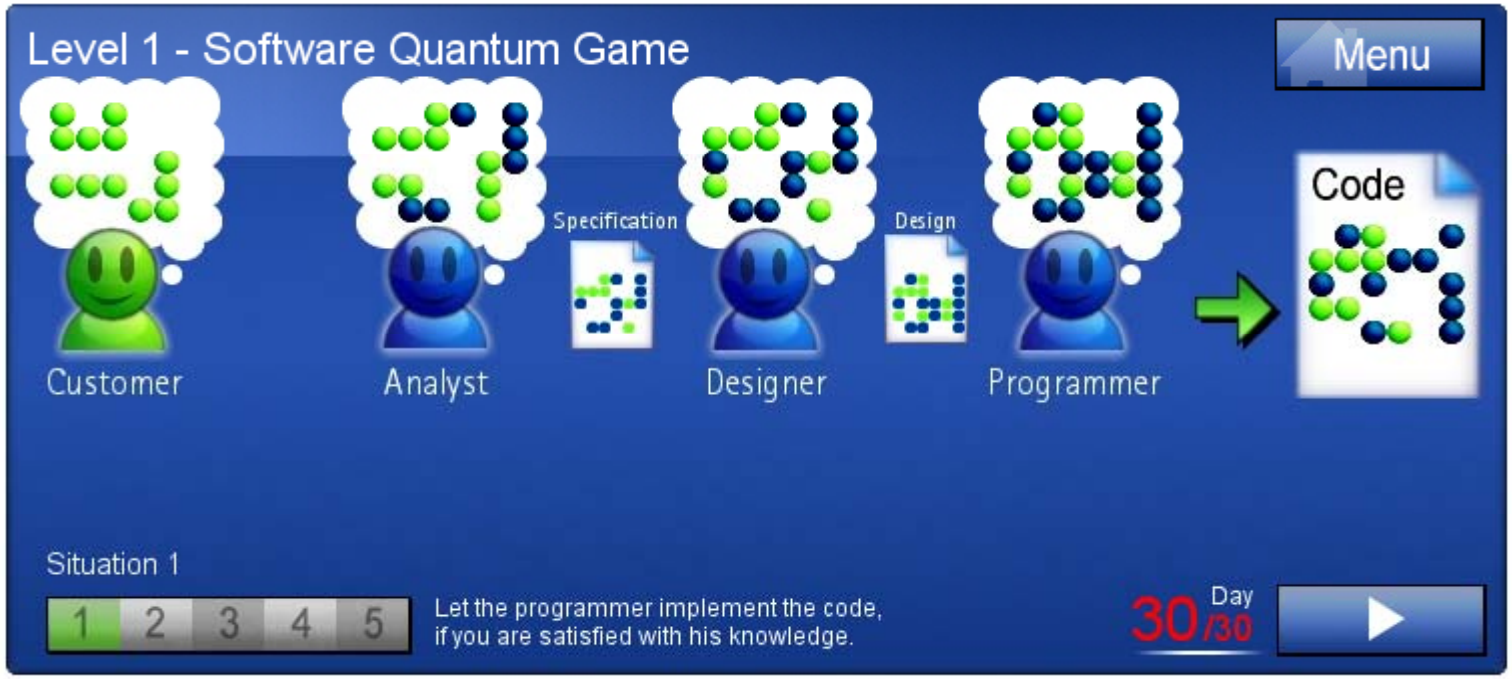
Related Work

- Abdel-Hamid, Madnick, *Software Project Dynamics*, 1991, Engelwood Cliffs, NJ: Prentice Hall
- Forrester, *Industrial dynamics*, 9. Edition, 1977 Cambridge.
- Rodriguez, Satpathy, Pfahl, *Effective software project management education through simulated models*. PROFES 2004
- Deininger, Schneider, *Teaching Software Project Management by Simulation – Experiences with a Comprehensive Model*, CSEE 1994

- No System Dynamics
- No support for Decision making
- No validation of underlying model through real projects
- Very limited number of aspects covered

→ Fast and simple Game

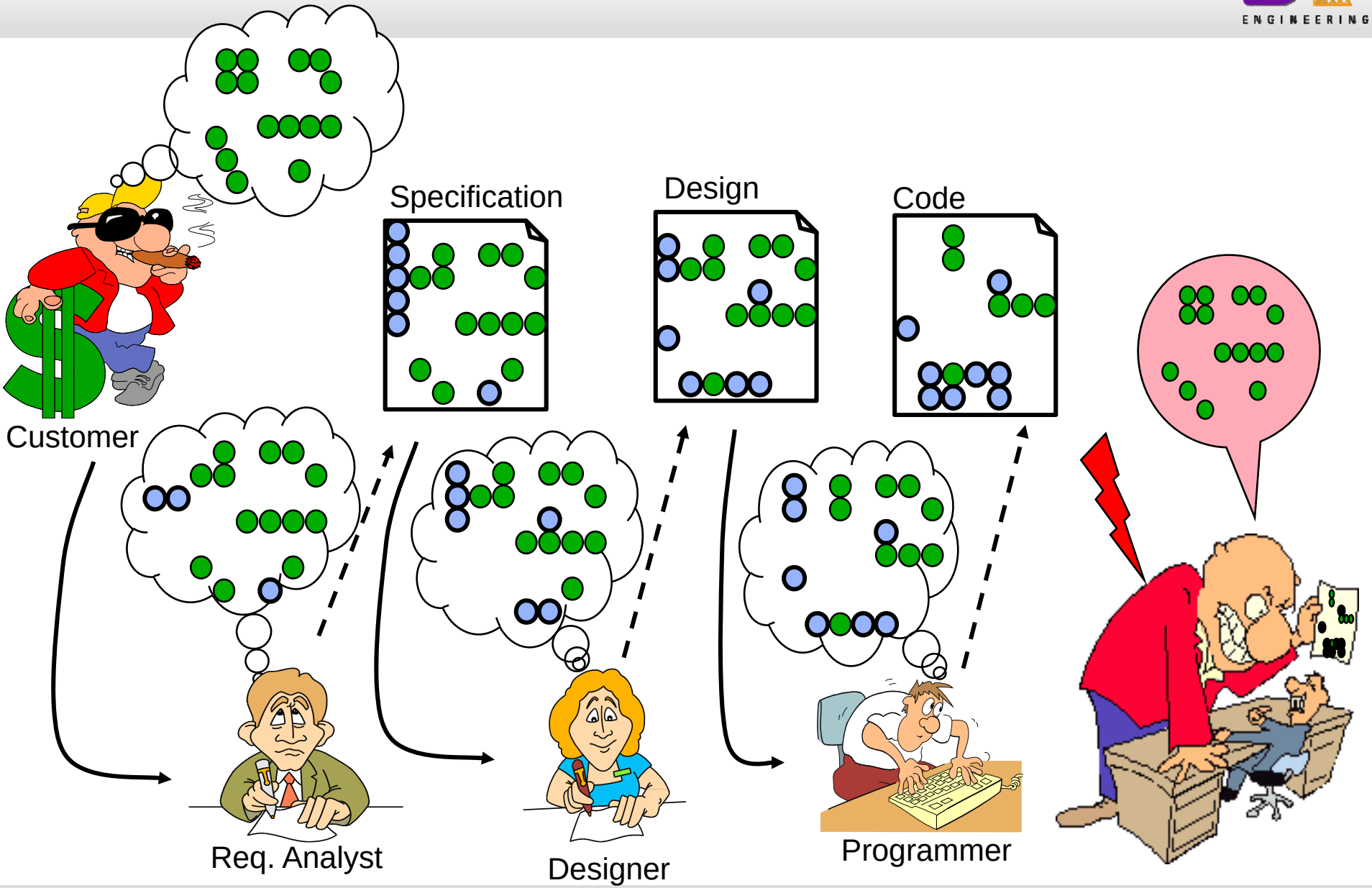
First Contact



It is your job to deliver a software (code) according to the customer's requirements in a time frame of 30 days. In level 1 you only have influence on the time of each communication, to fulfill the job: How long should the customer talk to the analyst? A click on the play button advances time for one day. But, use your time wisely! Otherwise the customer will not be satisfied with your work.

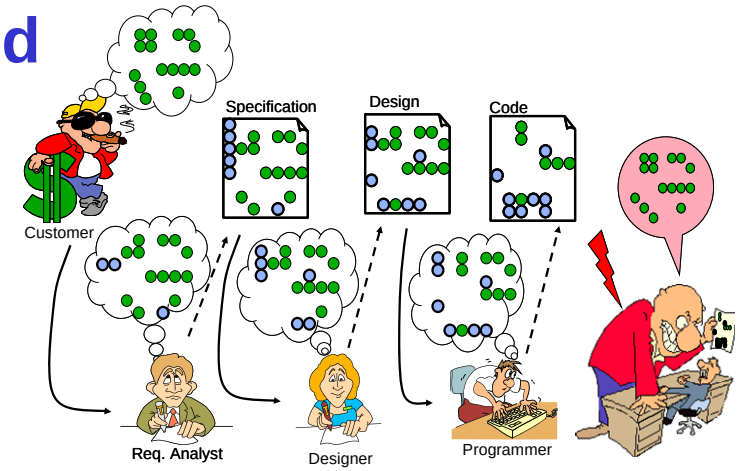
Hit the last arrow when you are ready to deliver the code to the customer. It is recommended to use all 30 days available before delivery. But, of course, you can finish the project early when clicking the play-button. When done, your customer will tell you if she was satisfied with your work.

Basic Concepts



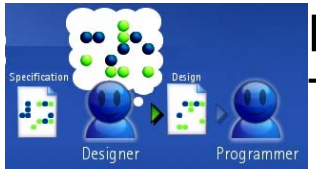
Software Quantum Axiomes

- Software Quantums are individuals, represented by the **unique position** of a ball
- Software quantums are anonymous. **All yellow balls** together model **all valid requirements** in the project.
- **Moving** one software quantum takes the same amount of time and effort, **independently of its color**.
- Colours of quantums are invisible to the simulated project participants. However, players can see them.
- **Reviews** are an opportunity to compare sets of software quantums, and **get rid of wrong ones**.
- Requirements are **right or wrong** (light or dark) only with respect to current **customer requirements**.



Goals

- Visualize a (limited) number of serious insights:

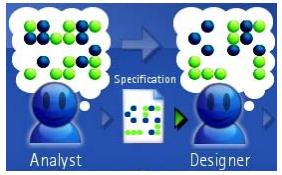


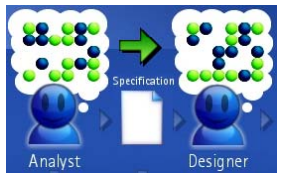
Flow of Requirements Through Projects

  - Alternate paths exist
  - Validation is important
  - Software Quality is defined by project priorities

- Intended Message:

Requirements that do not origin from customer are probably bad.

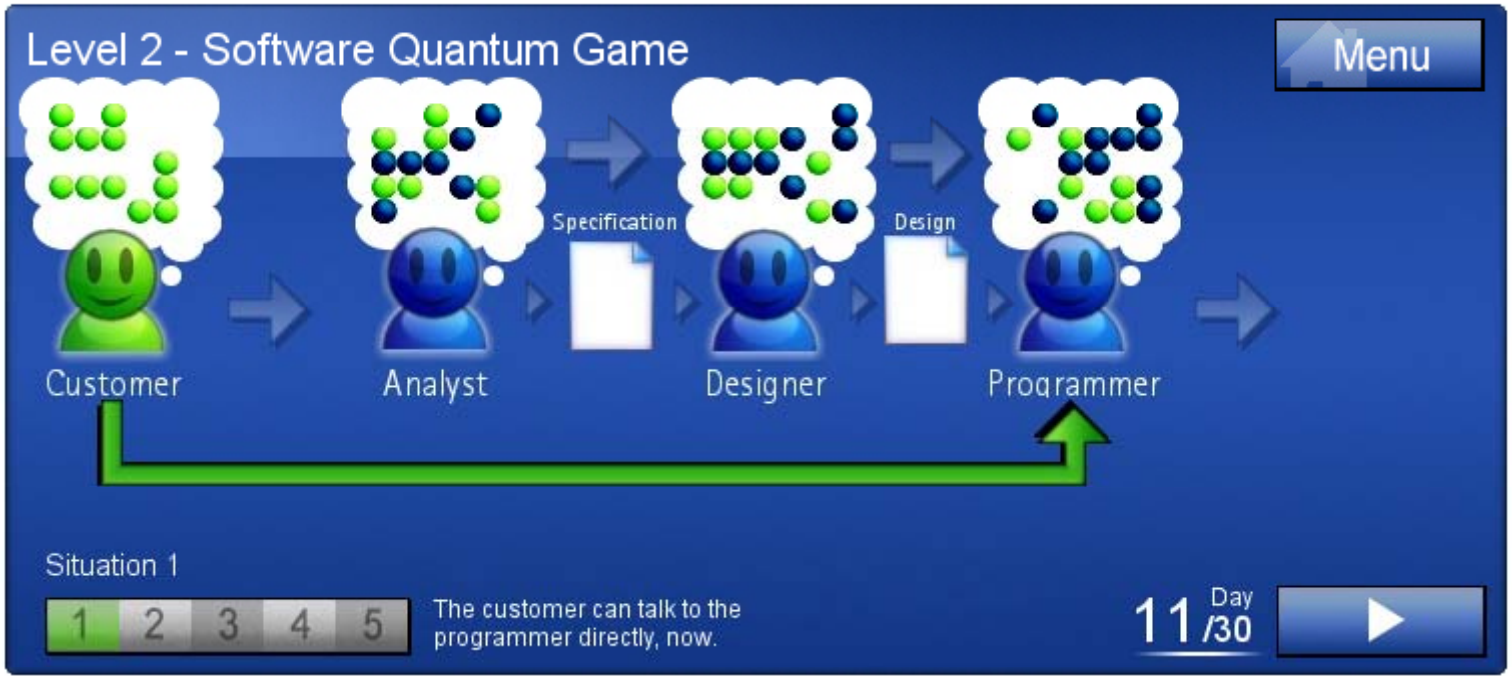




  - Document based
  - Direct communication
  - Misunderstandings will certainly exist
  - Small and urgent projects vs. safety critical projects

Game for Learning

http://www.se.uni-hannover.de/en/qgame

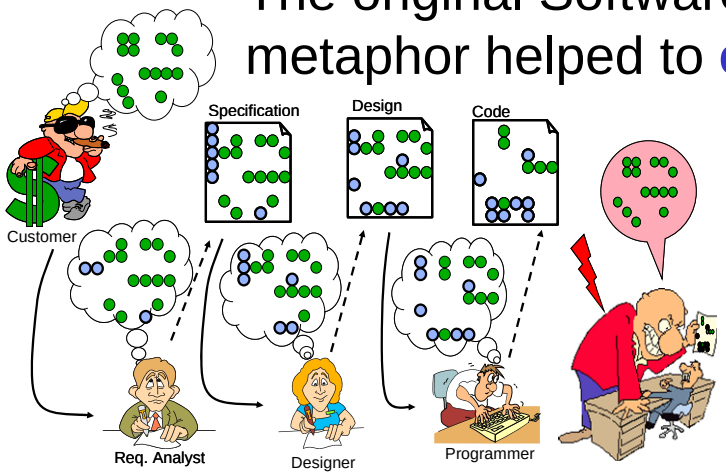


It is your job to deliver a software (code) according to the customer's requirements in a time frame of 30 days. In level 2 you not only have influence on the time of each communication, but you can also modify the order of talks and you can even skip documentation. A click on the play button advances time for one day. But, use your time wisely! Otherwise the customer will not be satisfied with your work.

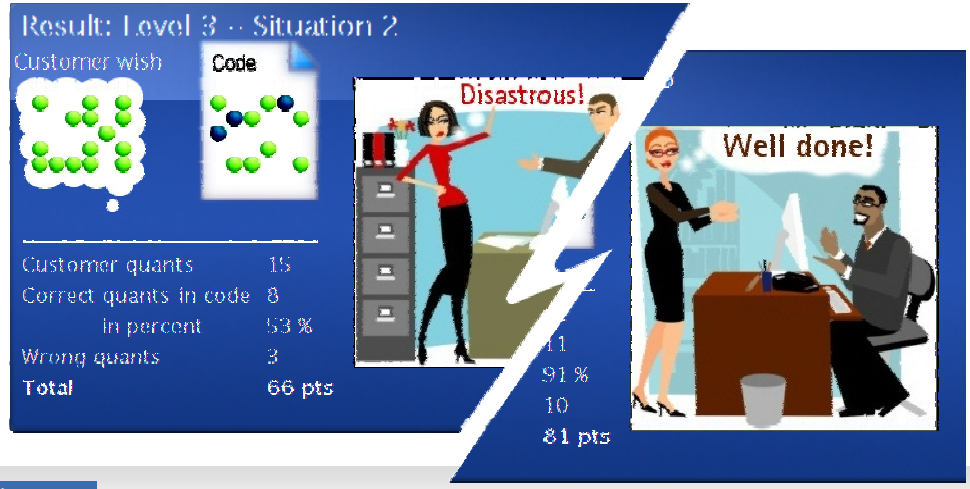
To establich communication between neighboring persons choose the arrow between them. If you want to modify the order of communication click on the arrows below the persons. Multiple clicks on the play-button affects communication time. By clicking on the last arrow you can finish the project at any time.

Conclusion

- The original Software Quantum metaphor helped to **explain RE**



- But before we start **discussion…**



- Making it interactive helps to **discuss RE**

Outlook

- Balancing
- Evaluation of teaching effect
- Evaluation of value in project planning
  - e.g. instructing project members,
    raise awareness of customers

http://www.se.uni-hannover.de/en/qgame
eric.knauss@inf.uni-hannover.de