

**The economics of software design (group discussion)**

1. Imagine you have to explain long-term impact of clean coding (and lack of it) to one of your prospect customers. Sketch a cost-per-change graph that visually conveys the message
2. "Incremental compromises can lead to failure" is a famous quote by Michael Nygard, the author of "Release It!" book. Can you explain it in terms of Drift to Low Performance phenomenon?
3. Discuss in groups: do you use SonarQube or any other tool for managing technical debt? If no, why and how can you start using that? If yes, what quality indicators do you watch?
4. How do you ensure that technical debt gets paid timely? Can you share any tips & tricks with the group?

**Naming and Nullity (work in pairs)**

1. In your own Gist, change a class so it represented entity, not a process (in other words, let a class be what it *really* *is*, not what it does). Make sure that accessors are *nouns* and mutators are *verbs*: [**http://bit.ly/jninja\_naming**](http://bit.ly/jninja_naming)
2. In your own Gist, improve the following code snippets: [**http://bit.ly/jninja\_refacto**r](http://bit.ly/jninja_refactor)