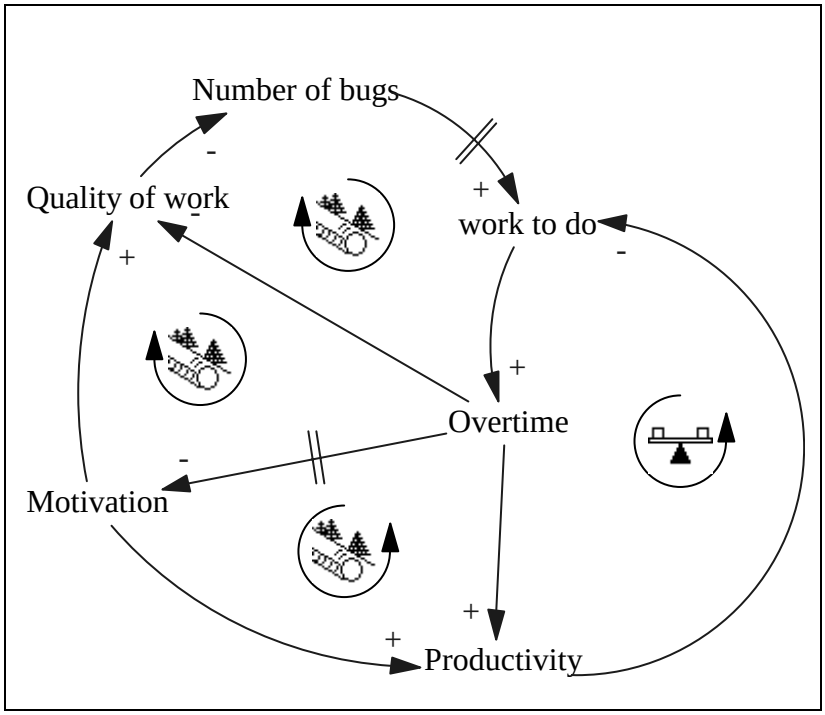
Together We Stand

Position Paper for the OOPSLA 2002 Workshop „Architecture is dead - long live the Architect“

Jens Coldewey
Coldewey Consulting
Curd-Jürgens-Str. 4
D-81739 München
Germany
Tel: +49-700-COLDEWEY
+49-700-26533939
Fax: +49-89-74995703
email: jens_coldewey@acm.org
http://www.coldewey.com

The literature has built up long lists of factors that are mission critical in a software project. Among them are stakeholder support or the ability to channel requirements and the client's expectations. To manage these issues is the task of the project manager. However, even with all these issues perfectly under control a project may fail for internal reasons. The classic example is a waterfall project that is not able to stabilize during debugging. If the test phase stretches infinitly, even the most generous (and well-managed) stakeholder eventually closes the faucet and kills the project.

In a systemic analysis of a project you try to identify non-linear cause-event chains, a task that usually leads feedback circles. The picture below is an example of such an analysis: If there is



a lot of work to do, many managers call for overtime, because they think that will raise productivity and thus reduce the work to do, as the right loop suggests. However, the work done in overtime usually is of poor quality and thus contain more bugs, which later leads to more bugs and therefore more work to do. This is shown in the upper left loop. A longer period of overtime

(more than one or two weeks) also lowers motivation, which again lowers productivity, and quality, both again leading to a higher workload. These relationships are non-linear, since there is no linear relationship between overtime, drop of motivation and quality. Rather, there are individual thresholds. If they are crossed, people may just quit or be so upset that it may take years to win them back, if you can do at all.

If you do a systemic analysis of a project, there are two major non-linear assets: Motivation of the team and changeability. If both are positive, most other internal obstacles can be dealt with. If one of them drifts off, the project will fail sooner or later. While motivation is the task of the project manager (or at least should be), changeability is responsibility of the architect.

Changeability covers two major aspects: The effort needed to change (or add) something to the system and the stability, that is the propability that you break existing parts of the features with the change. Both aspects are closely related, but it makes sense to separate them, because there are different techniques to ensure them. Separation of concerns is in my opinion the best of all bad tools we know to reduce the effort of changes, as long as it is combined with layers of abstraction. Where appropriate, I made excellent experiences with Test-first Programming to reduce the chance of breaking old code by adding new features (or debugging existing ones). Unfortunatly I couldn't find reasonable ways to do test-first with GUI programming or for time-critical, multi-session systems.

To ensure changeabiltiy is the major task of an architecture. Personally I found the academic definition of an architecture consisting of components and connectors not too helpful in this task. I'd rather understand the architecture as a picture, or vision of the system, much like Alan O'Callaghan's Modello pattern describes. It is the role of the architect to develop this vision together with the team and than keep up the flame. This is mainly a social task. The architect has to mediate between the different experts to help them find the best solution. The architecture has to be stable enough to provide a working backbone, yet felxible enough to adjust to changing requirements and to correct when it is found to be faulty. An architecture that doesn't change is a dead architecture and usually results in a dead project.

The architect has to closely collaborate with the project manager. Since the architect usually has the most accurate view on the technical status of the project, she or he can give the project manager early and often precise notice on risks lurking in the project. On the other hand the project manager relies on the architect's judgement when he or she has to estimate the complexity of requirements. Both roles are crucial in a project. Together they stand, divided they fall.