Towards using Extreme Programming to the Formal Specification of Software*

Hubert Baumeister

Institut für Informatik, LMU
baumeist@informatik.uni-muenchen.de

Lately an agile software development process called Extreme Programming (XP) has received a lot of attention. For small projects XP promises to produce high quality software that provides business value faster than traditional software processes. In addition, XP is designed to cope with changing requirements and to provide early feedback to cope with unclear requirements. On the other hand, people believe that XP is not very well suited to produce provably correct software because XP does not do up front specification and design.

When starting an XP-project the customer requirements are captured as so called user stories. Each user story describes a functionality of the system useful for the customer. The user stories are ordered by importance to the customer and estimated by the developers. While in traditional software development processes, first a design of the system is produced taking into account all requirements and then implemented, in XP the system is developed by taking a user story at a time starting from a birds eye view on the architecture called metaphor. For each new user story the simplest design is sought that incorporates the previous user stories and the new user story, and then this design is implemented. User stories not yet implemented do not contribute to the design. During this process it may be required to change the design and implementation of the existing functionality given by the previous user stories; this is called refactoring. To ensure that refactoring does not unintentionally destroy already implemented functionality, automatic tests are needed that check if the software still exhibits the desired functionality.

The advantage of this process is that it is easy to exchange requirements (user stories) as the software develops either due to changes in the business case or because of experiences gained with the already implemented software. Also higher quality software is produced as refactoring requires automatic tests with a good code coverage. Extreme Programming even advocates writing tests before writing code that make these tests run. This has the advantage that the programmer has to think about the functionality of the software before implementing it; further it ensures that no tests are forgotten.

The drawback of tests, however, is that they can only show that something is wrong and not that the software is correct. The idea that I want to present is that formal specifications are used instead of tests. However, in contrast to the traditional use of formal methods, not one big specification is produced before starting the implementation; instead, the specification is build incrementally and

* to be presented at WADT 2002.

in parallel with the software implementing the specification. In addition, tests can be automatically generated from these specifications to be used in refactorings.